Exhibit 14

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

CYWEE GROUP LTD., *Plaintiff,* v. GOOGLE, INC., *Defendant.*	CASE NO. **PLAINTIFF'S ORIGINAL COMPLAINT FOR PATENT INFRINGEMENT** JURY TRIAL DEMANDED

1. Plaintiff CyWee Group Ltd. ("Plaintiff" or "CyWee"), by and through its undersigned counsel, files this Original Complaint against Google, Inc. ("Google") as follows:

<u>THE PARTIES</u>

2. CyWee is a corporation existing under the laws of the British Virgin Islands with a principal place of business at 3F, No.28, Lane 128, Jing Ye 1st Road, Taipei, Taiwan 10462.

3. CyWee is a world-leading technology company that focuses on building products and providing services for consumers and businesses. CyWee has one of the most significant patent portfolios in the industry and is a market leader in its core development areas of motion processing, wireless high definition video delivery, and facial tracking technology.

4. Google, Inc. is a wholly-owned subsidiary of Alphabet, Inc. Google is a Delaware company with its principal place of business in California at 1600

Amphitheatre Parkway, Mountain View, CA 94043. Google may be served through its agent for service of process, Corporation Service Company, 2710 Gateway Oaks Drive, Suite 150N, Sacramento, CA 95833.

JURISDICTION AND VENUE

5. This action arises under the patent laws of the United States, 35 U.S.C. § 1 *et seq*. This Court has subject matter jurisdiction pursuant to 28 U.S.C. §§ 1331 and 1338(a).

6. This Court has personal jurisdiction over Google. Google has conducted and does conduct business within the State of Delaware. Google has purposefully and voluntarily availed itself of the privileges of conducting business in the United States and in the State of Delaware by continuously and systematically placing goods into the stream of commerce through an established distribution channel with the expectation that they will be purchased by consumers in Delaware. Google is registered to do business in Delaware and has authorized retailers for the accused products in this judicial district. Plaintiff's cause of action arises directly from Google's business contacts and other activities in the State of Delaware. Additionally, Google is incorporated in Delaware. Accordingly, this Court has personal jurisdiction over Google because it resides in this District.

7. Upon information and belief, Google has committed acts of infringement in this District giving rise to this action and does business in this

District, including making sales and/or providing service and support for their respective customers in this District. Google purposefully and voluntarily sold one or more of its infringing products with the expectation that they would be purchased by consumers in this District. These infringing products have been and continue to be purchased by consumers in this District. Google has committed acts of patent infringement within the United States and the State of Delaware.

8. Venue is proper as to Google under 28 U.S.C. § 1400(b) in that Google is incorporated in Delaware and, therefore, resides in this District.

BACKGROUND

Patentee And The Asserted Patents.

9. The Industrial Technology Research Institute ("ITRI") is a Taiwanese government- and industry-funded research and development center. In 2007, CyWee, which was started at ITRI, was formed. Its goal was to provide innovative motion-sensing technologies, such as those claimed in the patents-in-suit. Dr. Shun-Nan Liu and Chin-Lung Li, two of the inventors of the patents-in-suit, came to CyWee from ITRI. The third inventor, Zhou "Joe" Ye joined CyWee from private industry as its President and served as CEO from 2006 to 2016.

10. The inventors, Zhou Ye, Chin-Lung Li, and Shun-Nan Liou, conceived of the claims of the patents-in-suit—U.S. Patent No. 8,441,438 (the "'438 patent") and U.S. Patent No. 8,552,978 (the "'978 patent")—at CyWee

Group Ltd., located at 3F, No. 28, Lane 128, Jing Ye Road, Taipei. A true and correct copy of the ’438 patent and the ’978 patent are attached hereto as Exhibit A and Exhibit B respectively, in accordance with Local Rule 3.2.

11. Several claims of the patents-in-suit are entitled to a priority date of at least January 6, 2010 based on U.S. Provisional Application Serial No. 61/292,558, filed January 6, 2010 (“Provisional Application”).

12. Before May 22, 2009, CyWee began working on the “JIL Game Phone Project” or “JIL Phone.” Before July 29, 2009, CyWee developed a solution for the JIL Phone that practiced several claims of the ’438 patent. Those claims were diligently and constructively reduced to practice thereafter through the filing of the Provisional Application and were diligently and actually reduced to practice as discussed below. Accordingly, CyWee is entitled to a priority date of at least July 29, 2009 for several claims of the ’438 patent.

13. The JIL Phone was reduced to practice by at least September 25, 2009. The JIL Phone practiced several claims of both patents-in-suit. Accordingly, CyWee is entitled to a priority date of at least September 25, 2009 for several claims of the patents-in-suit.

Background Of The Technology.

14. The ’438 patent and ’978 patent are each directed to devices and methods for tracking the motion of a portable electronic device in 3D space and

compensating for accumulated errors to map the 3D movements of the device onto a display frame ('438 patent) or transform the 3D movements for a display, such as a 2D display on a computer or handheld device ('978 patent). '438 patent 1:17-52, 3:52-57; '978 patent 1:22-27, 7:5-18; **Exhibit C**, Declaration of Nicholas Gans, Ph.D. ("Gans Decl.") ¶ 8. At a high level, the patented inventions teach how to determine a device's current orientation based on motion data detected by its motion sensors, such as an accelerometer, gyroscope, and magnetometer. '438 patent 4:6-30; '978 patent 4:15-44; Gans Decl. ¶ 8. The '438 patent and '978 patent describe portable electronic devices or pointing devices such as smartphones and navigation equipment. '978 patent 22:34-40, Fig. 6; '438 patent 4:6-30, Fig. 6; Gans. Decl. ¶ 8.

15. There are different types of motion sensors, including accelerometers, gyroscopes, and magnetometers. Gans Decl. ¶ 9. Accelerometers measure accelerations. *Id.* For example, airbags use accelerometers, such that the airbag is triggered based on sudden deceleration. Accelerometers can also measure forces due to gravity. *Id.* Gyroscopes measure rotation rates or angular velocities. *Id.* Magnetometers measure magnetism, including the strength of a magnetic field along a particular direction. *Id.* Each type of motion sensor is subject to inaccuracies. *Id.* For example, a gyroscope sensor has a small, added offset or bias. *Id.* This bias will accumulate over time and lead to large drift error. *Id.* Similarly,

magnetometers are subject to interference from natural and manmade sources (e.g., power electronics). *Id.* Additionally, errors can accumulate over time. *Id.* These sensors typically take measurements along a single direction. *Id.* To accurately measure motions along an arbitrary axis, three like sensors are grouped together and aligned at right angles. *Id.* Such a sensor set is generally referred to as a 3-axis sensor. *Id.*

16. Orientation information returned by the claimed inventions of the '438 patent and '978 patent has many uses, particularly for mobile cellular devices, such as navigation, gaming, and augmented/virtual reality applications. Gans Decl. ¶ 12. Navigation applications can use orientation information to determine the heading of the phone, indicate what direction the user is facing, and automatically orient the map to align with the cardinal directions. *Id.* Increasing numbers of games and other applications use the motion of the phone to input commands, such as tilting the mobile device like a steering wheel. *Id.* Augmented and virtual reality applications rely on accurate estimation of the device orientation in order to render graphics and images at the proper locations on the screen. *Id.*

17. Prior to 2010, motion sensors had limited applicability to portable electronic devices due to a variety of technological hurdles. Gans Decl. ¶ 13. For example, different types of acceleration (e.g., linear, centrifugal, gravitational) could not be readily distinguished from one another, and rapid, dynamic, and

unexpected movements caused significant errors and inaccuracies. *Id.* These difficulties were compounded by the miniaturization of the sensors necessary to incorporate them in portable electronic devices. *Id.* With the development of micro-electromechanical systems, or "MEMS," miniaturized motion sensors could be manufactured and incorporated on a semiconductor chip, but such MEMS sensors had significant limitations. *Id.*

18. For example, it is impossible for MEMS accelerometers to distinguish different types of acceleration (e.g., linear, centrifugal, gravitational). Gans Decl. ¶ 14. When a MEMS accelerometer is used to estimate orientation, it must measure force along the direction of gravity (i.e., down), but that gravitational measurement can be "interfused" with other accelerations and forces (e.g., vibration or movement by the person holding the device). *Id.* Thus, non-gravitational accelerations and forces must be estimated and subtracted from the MEMS accelerometer measurement to yield an accurate result. *Id.* A MEMS gyroscope is prone to drift, which will accumulate increasing errors over time if not corrected by another sensor or recalibrated. *Id.* A MEMS magnetometer is highly sensitive to not only the earth's magnetic fields, but other sources of magnetism (e.g., power lines and transformers) and can thereby suffer inaccuracies from environmental sources of interference that vary both in existence and intensity from location to location. *Id.*

19. Additionally, orientation cannot be accurately calculated using only one type of MEMS sensor. Gans Decl. ¶ 15. For example, if only a 3-axis MEMS accelerometer is used to measure orientation, pitch and yaw can be measured, but not roll. *Id.* If only a MEMS gyroscope is used to measure angular velocity, only relative changes in orientation can be measured, not absolute orientation. *Id.*

20. Without orientation information, mobile device apps would be limited to very static operation. Gans Decl. ¶ 16. This was the scenario with initial smart phones and other mobile devices. *Id.* Navigation aids could render a map and indicate the location of the device using GPS. *Id.* However, these maps would orient with North on the map pointing to the top of the screen. *Id.* The user could rotate the map using touch commands, but the map would not rotate automatically as the user turned. *Id.* Nor could the device indicate what direction the device was facing. *Id.*

21. Many games use motion of the device to control the game. Gans Decl. ¶ 17. A common control scheme, especially for driving and piloting games, is to have the user rotate the device, such as a phone or game controller, like a steering wheel to indicate the direction the vehicle should move. *Id.* Some puzzle games also use motions to cause elements of the game to move. *Id.* As discussed previously, accelerometers measure acceleration, which is a very noisy signal. *Id.* Acceleration is the derivative of velocity, which is the derivative of position. *Id.*

Small magnitude noise can have large derivatives, which means that small levels of noise from vibration or electrical fluctuations will be magnified at the acceleration level. *Id.* Even a stationary device will have notable noise measured by an accelerometer. *Id.* A moving device will only amplify this noise. *Id.* Since accelerometers measure linear and centripetal accelerations as well as the acceleration of gravity, orientation estimates on a moving device will not be accurate. *Id.*

22. If only an accelerometer is used, a coarse estimate of the device orientation can be obtained by averaging or numerically filtering the results. Gans Decl. ¶ 18. Essentially, the device can determine if it is tilted left or right, up or down, but the exact angle cannot be estimated accurately while in motion. *Id.* This is suitable for games to move a character or steer a vehicle in a particular direction, but generally cannot utilize the magnitude of tilt to move at corresponding faster or slower speeds. *Id.*

23. Movement on a display can, of course, be controlled by means other than a portable electronic device with orientation sensors. Gans Decl. ¶ 19. For example, games could be controlled using traditional "joystick" type inputs. *Id.* For smart phones with touch screens, commands are given by having the user touch specific parts of the screen. *Id.*

24. For other current applications, portable electronic devices with orientation sensors are more crucial. Gans Decl. ¶ 20. Augmented reality (AR) and virtual reality (VR) are new and growing classes of applications for smart phones and mobile devices. *Id.* In AR, the device camera provides live video feed to the screen, and the application overlays generate graphics onto the screen at specific locations. *Id.* AR navigation apps can draw signs or labels to indicate what specific places or objects are or can render arrows or other indicators. *Id.* AR games and teaching applications can label objects or draw characters or items such that they appear as if they are in the real world seen in the video. *Id.* Virtual reality is similar but does not use the camera, rather it completely renders an artificial 3D environment on the screen. *Id.* VR most often requires a head set such that the user only sees the screen. *Id.* Mobile devices and smart phones used for VR generally split the screen and display to two side-by-side images of the rendered environment that are slightly offset to simulate a left and right eye. *Id.* The device then sits in a headset with lenses such that the user has each eye see only one of the split-screen images and has a sense of stereo (3D) vision. *Id.*

25. Without orientation sensing, AR and VR applications cannot work. Gans Decl. ¶ 21. The system will have no ability to understand the orientation of the device and know where to draw objects and/or the scene. *Id.* The rough orientation estimate provided by an accelerometer (ideally with a magnetometer)

will not be sufficient to track during typical head motions. *Id.* It has been demonstrated that VR applications that use an accelerometer often cause motion sickness, as the rendered images do track with the head motions. *Id.* An AR application with the use of a gyroscope and fusion algorithm will not render objects at the correct locations, and may obscure the view rather than provide helpful information. *Id.*

26. There are ways to estimate orientation other than the approaches presented in the ’438 patent or ’978 patent, which involve algorithms that filter and fuse measurements from inertial and magnetic sensors. Gans Decl. ¶ 22. Most such methods are based on cameras and computer vision algorithms. *Id.* However, the limitations of these methods render them unusable for portable electronic devices. *Id.* For example, there are a variety of motion capture systems that use cameras arrayed around an environment. *Id.* Markers (e.g., reflective balls) can be placed on objects, and the cameras can locate the markers, often to sub-mm accuracy. *Id.* If an object has three or more markers on it, the orientation of the object can be determined with sub-degree accuracy. *Id.* This method is very accurate, but quite expensive (often about $100,000). *Id.* The cameras are fixed in place, and the estimation can only work within a small space (a box of dimensions on the order of tens of meters). *Id.* This is not suitable for the vast majority of mobile device users or applications. *Id.*

27. A camera on a portable electronic device, such as a smart phone, can be used to estimate orientation of the phone. Gans Decl. ¶ 23. One class of approaches to this problem uses special patterns or markers in the environment. *Id.* These often have the appearance of a QR code or 2D UPC. *Id.* Taking a picture of the pattern, computer vision algorithms can determine the position and orientation of the camera with respect to the marker. *Id.* AR applications have placed the patterns on specific objects or consumer products so the device can render images and graphics with respect to the pattern. *Id.* AR games have included patterned mats that are placed on a table or other flat surface, and the device renders characters and objects as if they were on the surface. *Id.*

28. Multiple unique patterns can be placed around an environment; so long as one is always in view, the camera can maintain an estimate of the orientation and position. Gans Decl. ¶ 24. In this way, it can be used for navigation. *Id.* The necessity of placing patterns would make this approach useless for a majority of applications, particularly outdoors. *Id.* The camera would also need to remain on at all times, which would cause severe battery drain. *Id.*

29. Orientation of the camera can also be estimated over an indefinite amount of time using vision algorithms known as visual odometry. Gans Decl. ¶ 25. In visual odometry, changes in the image over time are used to estimate the camera velocity. *Id.* This velocity can be integrated over time to estimate the

change in orientation. While these methods are well understood, they can only track change in relative orientation, not give absolute orientation. *Id.* They also require the camera to be on at all times, which will greatly reduce battery life. *Id.*

<u>The Prior Art.</u>

30. As noted in both the ’438 patent and ’978 patent, prior art portable electronic devices, such as pointing devices, smartphones and navigation equipment, had several deficiencies in addressing the technological challenges of mapping and transforming movement in a 3D space to a 2D display. These prior art portable devices could only output the movement of the device in 2D, rather than the 3D reference frame of the ’438 and’978 patents. ’438 patent 2:47-55; ’978 patent 2:41-58. In addition, the portable devices could not accurately calculate and account for movements of the device in a dynamic environment, such as erroneous drift measurements of the device or accelerations along with the direction of gravity. ’438 patent 2:55-62; ’978 patent 2:58-66. These prior art portable devices were also limited to detecting gravitational acceleration detected by the accelerometer and were therefore incapable of accurately outputting the actual yaw, pitch and roll angles. ’438 patent 2:62-3:5; ’978 patent 2:66-3:13. Finally, for the specific case of pointing devices, when they extended beyond the border or boundary of the display, the absolute movement pattern was not mapped, but instead the location outside the boundary was ignored and a relative movement

pattern used, which resulted in uncompensated errors. ’438 patent 3:16-51; ’978 patent 3:20-52.

<u>PATENT INFRINGEMENT OF U.S. PATENT NO. 8,441,438</u>

31. Plaintiff repeats and re-alleges each and every allegation of paragraphs 1-30 as though fully set forth herein.

32. The ’438 patent, titled “3D Pointing Device and Method for Compensating Movement Thereof,” was duly and legally issued by the United States Patent and Trademark Office on May 14, 2013 to CyWee Group Limited, as assignee of named inventors Zhou Ye, Chin-Lung Li, and Shun-Nan Liou.

33. CyWee is the owner of all right, title, and interest in and to the ’438 patent with full right to bring suit to enforce the patent, including the right to recover for past infringement damages.

34. The ’438 patent claims, *inter alia*, a machine capable of detecting, measuring, and calculating the movements and rotations of the machine—utilizing, *inter alia*, a six-axis motion sensor module, a data transmitting unit, and a computing processor in one or more claimed configurations—and methods for measuring and calculating the movements and rotations of a device within a spatial reference frame. *See, generally,* Gans Decl.¶¶ 8-12.

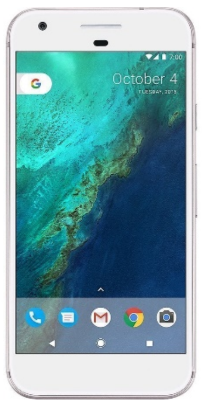
35. The ’438 patent is directed to useful and novel particular embodiments and methods for detecting, measuring, and calculating motion within

a spatial reference frame. *Id.* ¶ 27. Specifically, the ’438 patent claims a novel system involving multiple sensor types and a novel method for using those sensors to overcome the limitations of the individual sensor types in accurately determining the orientation of a device. *See id.* ¶¶ 26-28. The ’438 patent is not intended to, and does not claim every possible means of detecting, measuring, and calculating motion within a spatial reference frame. There are alternative methods to determining orientation within a spatial reference frame, such as systems and methods utilizing computer vision algorithms and/or cameras. *See id.* ¶¶ 22-25, 33. The ’438 patent is directed to a technological solution to a technological problem. *Id.* ¶¶ 33-35. Accordingly, the ’438 patent is not directed to, and does not claim, the mere concept of motion sensing or of detecting, measuring, and calculating motion within a spatial reference frame. *Id.* ¶¶ 29-35.

36. Each and every claim of the ’438 patent is valid and enforceable and each enjoys a statutory presumption of validity separate, apart, and in addition to the statutory presumption of validity enjoyed by every other of its claims. 35 U.S.C. § 282.

37. CyWee is informed and believes, and thereupon alleges, that Google has been, and is currently, directly and/or indirectly infringing one or more claims of the ’438 patent in violation of 35 U.S.C. § 271, including as stated below.

38. CyWee is informed and believes, and thereupon alleges, that Google has directly infringed, literally and/or under the doctrine of equivalents, and will continue to directly infringe claims of the '438 patent by making, using, selling, offering to sell, and/or importing into the United States products that embody or practice the apparatus and/or method covered by one or more claims of the '438 patent including, but not limited to, Google's following devices:



Google Pixel

Google Pixel 2

Google Pixel XL

Google Pixel 2 XL

The foregoing devices are collectively referred to as the "'438 Accused Products" and include the below specifications and features.

39. On information and belief, Google indirectly infringes the '438 patent by inducing others to infringe one or more claims of the '438 patent through sale and/or use of the '438 Accused Products. On information and belief, at least as a result of the filing of this action, Google is aware of the '438 patent; is aware that its actions with regards to distributors, resellers, and/or end users of the '438 Accused Products would induce infringement; and despite such awareness will

continue to take active steps—such as, creating and disseminating the ’438 Accused Products and product manuals, instructions, promotional and marketing materials, and/or technical materials to distributors, resellers, and end users—encouraging others’ infringement of the ’438 patent with the specific intent to induce such infringement.[1] Users of the ’438 Accused Products infringe through the ordinary use of such products. Google further induces others to infringe by marketing the accused products VR and/or AR capabilities. *See* https://store.google.com/ us/product/pixel_2?utm_source=en-ha-na-sem&utm_medium=text&utm_term=US sale&ds_kid=43700024791423013&utm_content=bkws&utm_campaign=Pixel&gclid=Cj0KCQiAyszSBRDJARIsAHAqQ4ouuVPL5FfpJ6LKd-eFT_H963mAond-6M5FDxrDHnEZrtiKvPQsGFwaAqRGEALw_wcB&gclsrc=aw.ds&dclid=CPCzr7D8yNgCFYYC0wodJwwB-Q (“Better with Pixel High quality VR, wherever you go.”); *see also* https://store.google.com/us/product/google_daydream_view (describing “Google Daydream View.”). CyWee expects discovery to provide more facts relevant to Google’s induced infringement.

40. The Google Pixel includes a display screen.

[1] To preempt any argument that such allegations are insufficient to establish a claim for induced infringement, CyWee respectfully notes that at least one court previously held such allegations sufficient. *See, e.g., Huawei Techs. Co. v. T-Mobile US, Inc.*, Case No. 2:16-cv-00052-JRG-RSP, 2017 WL 1129951, at *3 (E.D. Tex. Feb. 21, 2017) (“Huawei’s complaints adequately plead knowledge. Huawei alleges that T-Mobile knew of the asserted patents ‘since at least the filing of this action.’”).

41. The Google Pixel includes a housing.

42. The Google Pixel includes a 3-axis accelerometer.

43. The Google Pixel includes a 3-axis gyroscope.

44. The Google Pixel includes at least one printed circuit board ("PCB").

45. The Google Pixel includes a 3-axis accelerometer attached to a PCB.

46. The Google Pixel includes a 3-axis gyroscope attached to a PCB.

47. The Google Pixel includes a 3-axis accelerometer that is capable of measuring accelerations.

48. The Google Pixel includes a 3-axis gyroscope that is capable of measuring rotation rates.

49. The Google Pixel runs an Android^{TM} operating system.

50. The Google Pixel includes a 3-axis accelerometer that is capable of measuring accelerations using a "Sensor Coordinate System" as described in the Android^{TM} developer library. *See* https://developer.android.com/guide/topics/sensors/sensors_overview.html (describing "Sensor Coordinate System").

51. The Google Pixel includes a 3-axis gyroscope that is capable of measuring rotation rates using a "Sensor Coordinate System."

52. The Google Pixel includes a processor that is capable of processing data associated with measurement from a 3-axis accelerometer.

53. The Google Pixel includes a processor that is capable of processing data associated with measurement from a 3-axis gyroscope.

54. The Android™ operating system that runs on the Google Pixel uses the measurement from a 3-axis accelerometer included in the device.

55. The Android™ operating system that runs on the Google Pixel uses the measurement from a 3-axis gyroscope included in the device.

56. The Android™ operating system that runs on the Google Pixel uses the measurement from a 3-axis accelerometer and the measurement from a 3-axis gyroscope to calculate an attitude of the device.

57. The Google Pixel 2 includes a display screen.

58. The Google Pixel 2 includes a housing.

59. The Google Pixel 2 includes a 3-axis accelerometer.

60. The Google Pixel 2 includes a 3-axis gyroscope.

61. The Google Pixel 2 includes at least one PCB.

62. The Google Pixel 2 includes a 3-axis accelerometer attached to a PCB.

63. The Google Pixel 2 includes a 3-axis gyroscope attached to a PCB.

64. The Google Pixel 2 includes a 3-axis accelerometer that is capable of measuring accelerations.

65. The Google Pixel 2 includes a 3-axis gyroscope that is capable of measuring rotation rates.

66. The Google Pixel 2 runs an Android™ operating system.

67. The Google Pixel 2 includes a 3-axis accelerometer that is capable of measuring accelerations using a "Sensor Coordinate System" as described in the Android™ developer library. *See* https://developer.android.com/guide/topics/sensors/sensors_overview.html (describing "Sensor Coordinate System").

68. The Google Pixel 2 includes a 3-axis gyroscope that is capable of measuring rotation rates using a "Sensor Coordinate System."

69. The Google Pixel 2 includes a processor that is capable of processing data associated with measurement from a 3-axis accelerometer.

70. The Google Pixel 2 includes a processor that is capable of processing data associated with measurement from a 3-axis gyroscope.

71. The Android™ operating system that runs on the Google Pixel 2 uses the measurement from a 3-axis accelerometer included in the device.

72. The Android™ operating system that runs on the Google Pixel 2 uses the measurement from a 3-axis gyroscope included in the device.

73. The Android™ operating system that runs on the Google Pixel 2 uses the measurement from a 3-axis accelerometer and the measurement from a 3-axis gyroscope to calculate an attitude of the device.

74. The Google Pixel XL includes a display screen.

75. The Google Pixel XL includes a housing.

76. The Google Pixel XL includes a 3-axis accelerometer.

77. The Google Pixel XL includes a 3-axis gyroscope.

78. The Google Pixel XL includes at least one PCB.

79. The Google Pixel XL includes a 3-axis accelerometer attached to a PCB.

80. The Google Pixel XL includes a 3-axis gyroscope attached to a PCB.

81. The Google Pixel XL includes a 3-axis accelerometer that is capable of measuring accelerations.

82. The Google Pixel XL includes a 3-axis gyroscope that is capable of measuring rotation rates.

83. The Google Pixel XL runs an Android™ operating system.

84. The Google Pixel XL includes a 3-axis accelerometer that is capable of measuring accelerations using a "Sensor Coordinate System" as described in the Android™ developer library. *See* https://developer.android.com/guide/topics/sensors/sensors_overview.html (describing "Sensor Coordinate System").

85. The Google Pixel XL includes a 3-axis gyroscope that is capable of measuring rotation rates using a "Sensor Coordinate System."

86. The Google Pixel XL includes a processor that is capable of processing data associated with measurement from a 3-axis accelerometer.

87. The Google Pixel XL includes a processor that is capable of processing data associated with measurement from a 3-axis gyroscope.

88. The Android™ operating system that runs on the Google Pixel XL uses the measurement from a 3-axis accelerometer included in the device.

89. The Android™ operating system that runs on the Google Pixel XL uses the measurement from a 3-axis gyroscope included in the device.

90. The Android™ operating system that runs on the Google Pixel XL uses the measurement from a 3-axis accelerometer and the measurement from a 3-axis gyroscope to calculate an attitude of the device.

91. The Google Pixel 2 XL includes a display screen.

92. The Google Pixel 2 XL includes a housing.

93. The Google Pixel 2 XL includes a 3-axis accelerometer.

94. The Google Pixel 2 XL includes a 3-axis gyroscope.

95. The Google Pixel 2 XL includes at least one PCB.

96. The Google Pixel 2 XL includes a 3-axis accelerometer attached to a PCB.

97. The Google Pixel 2 XL includes a 3-axis gyroscope attached to a PCB.

98. The Google Pixel 2 XL includes a 3-axis accelerometer that is capable of measuring accelerations.

99. The Google Pixel 2 XL includes a 3-axis gyroscope that is capable of measuring rotation rates.

100. The Google Pixel 2 XL runs an Android™ operating system.

101. The Google Pixel 2 XL includes a 3-axis accelerometer that is capable of measuring accelerations using a "Sensor Coordinate System" as described in the Android™ developer library. *See* https://developer.android.com/guide/topics/sensors/sensors_overview.html (describing "Sensor Coordinate System").

102. The Google Pixel 2 XL includes a 3-axis gyroscope that is capable of measuring rotation rates using a "Sensor Coordinate System."

103. The Google Pixel 2 XL includes a processor that is capable of processing data associated with measurement from a 3-axis accelerometer.

104. The Google Pixel 2 XL includes a processor that is capable of processing data associated with measurement from a 3-axis gyroscope.

105. The Android™ operating system that runs on the Google Pixel 2 XL uses the measurement from a 3-axis accelerometer included in the device.

106. The Android™ operating system that runs on the Google Pixel 2 XL uses the measurement from a 3-axis gyroscope included in the device.

107. The Android^{TM} operating system that runs on the Google Pixel 2 XL uses the measurement from a 3-axis accelerometer and the measurement from a 3-axis gyroscope to calculate an attitude of the device.

108. Google's actions with regards to distributors, resellers, and/or end users of the '438 Accused Products induce infringement of the patent by others, and at least as a result of the filing of this Complaint, Google is aware that its actions induce infringement. Despite such awareness, upon information and belief, Google will continue to take active steps—such as creating and disseminating the '438 Accused Products, accessories thereto, product manuals, instructions, support materials, promotional and marketing materials, and/or technical materials to distributors, resellers, and end users—encouraging others to infringe the '438 Patent with the specific intent to induce such infringement.

109. Google provides manuals and/or instructions for the '438 Accused Products and/or provides instructional and support materials on its website that teach and instruct its customers to operate those products in ways that practice the claimed invention. For instance, Google provides support documents for the '438 Accused Products that instruct users on how to use motion gestures to perform various device functions. *See, e.g.*, https://support.google.com/pixelphone/answer/7 443425?hl=en (last visited April 4, 2018) (describing how to check device notifications by lifting a '438 Accused

Product and how to switch between front and rear facing cameras using a "double-twist gesture").

110. Additionally, Google offers the Google Daydream View as an accessory for use with the '438 Accused Products and other products that practice the '438 patent and specifically advertises and instructs users in how to use a '438 Accused Product with the Google Daydream View in a manner that infringes the '438 patent. *See* https://store.google.com/product/google_daydream_view (last visited April 4, 2018); *see also* https://store.google.com/product/google_daydream_ view_learn (last visited April 4, 2018) (describing the "Swivel and Play" feature of the Google Daydream View and instructing users to "[u]se a swivel chair to spin 360° and see all around you.").

111. These support materials and accessories, among others, demonstrate that Google actively induces users to operate their products in ways that practice the claimed invention. Upon information and belief, Google will continue to instruct end-users of the '438 Accused Products to operate those products in ways that practice the claimed invention even after being put on actual notice of the infringement of the '438 Patent.

112. CyWee adopts, and incorporates by reference, as if fully stated herein, the attached claim chart for claim 14 of the '438 patent, which is attached hereto as **Exhibit D**. The claim chart describes and demonstrates how Google infringes the

’438 patent. In addition, CyWee alleges that Google infringes one or more additional claims of the ’438 patent in a similar manner.

113. Google’s acts of infringement have caused and will continue to cause substantial and irreparable damage to CyWee.

114. As a result of Google’s infringement of the ’438 patent, CyWee has been damaged. CyWee is, therefore, entitled to damages pursuant to 35 U.S.C. § 284 in an amount that presently cannot be pled but that will be determined at trial.

PATENT INFRINGEMENT OF U.S. PATENT NO. 8,552,978

115. Plaintiff repeats and re-alleges each and every allegation of paragraphs 1-114 as though fully set forth herein.

116. The ’978 patent, titled “3D Pointing Device and Method for Compensating Rotations of the 3D Pointing Device Thereof,” was duly and legally issued by the United States Patent and Trademark Office on October 8, 2013 to CyWee Group Limited, as assignee of named inventors Zhou Ye, Chin-Lung Li, and Shun-Nan Liou.

117. CyWee is the owner of all right, title, and interest in and to the ’978 patent with full right to bring suit to enforce the patent, including the right to recover for past infringement damages.

118. The ’978 patent claims, *inter alia*, a machine capable of detecting, measuring, and calculating the movements and rotations of the machine—utilizing,

inter alia, a nine-axes motion sensor module and two computing processors in one or more claimed configurations—and methods for measuring and calculating the movements and rotations of a device within a spatial reference frame. *See, generally,* Gans Decl.¶¶ 8-12.

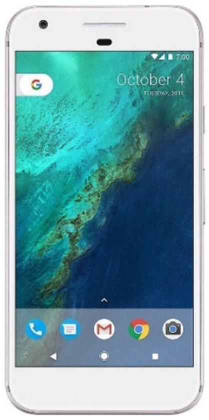
119. The ’978 patent is directed to useful and novel particular embodiments and methods for detecting, measuring, and calculating motion within a spatial reference frame. *Id.* ¶ 27. Specifically, the ’978 patent claims a novel system involving multiple sensor types and a novel method for using those sensors to overcome the limitations of the individual sensor types in accurately determining the orientation of a device. *See id.* ¶¶ 26-28. The ’978 patent is not intended to, and does not claim every possible means of detecting, measuring, and calculating motion within a spatial reference frame. There are alternative methods to determining orientation within a spatial reference frame, such as systems and methods utilizing computer vision algorithms and/or cameras. *See id.* ¶¶ 22-25, 33. The ’978 patent is directed to a technological solution to a technological problem. *Id.* ¶¶ 33-35. Accordingly, the ’978 patent is not directed to, and does not claim, the mere concept of motion sensing or of detecting, measuring, and calculating motion within a spatial reference frame. *Id.* ¶¶ 29-35.

120. Each and every claim of the ’978 patent is valid and enforceable and each enjoys a statutory presumption of validity separate, apart, and in addition to

the statutory presumption of validity enjoyed by every other of its claims. 35 U.S.C. § 282.

121. CyWee is informed and believes, and thereupon alleges, that Google has been, and is currently, directly and/or indirectly infringing one or more claims of the ’978 patent in violation of 35 U.S.C. § 271, including as stated below.

122. CyWee is informed and believes, and thereupon alleges, that Google has directly infringed, literally and/or under the doctrine of equivalents, and will continue to directly infringe claims of the ’978 patent by making, using, selling, offering to sell, and/or importing into the United States products that embody or practice the apparatus and/or method covered by one or more claims of the ’978 patent including, but not limited to, Google’s following devices:



Google Pixel

Google Pixel 2

Google Pixel XL

Google Pixel 2 XL

123. The foregoing devices are collectively referred to as the "'978 Accused Products" and include the following specifications and features.

124. On information and belief, Google indirectly infringes the '978 patent by inducing others to infringe one or more claims of the '978 patent through sale and/or use of the '978 Accused Products. On information and belief, at least as a result of the filing of this action, Google is aware of the '978 patent; is aware that its actions with regards to distributors, resellers, and/or end users of the '978 Accused Products would induce infringement; and despite such awareness will continue to take, active steps—such as, creating and disseminating the '978 Accused Products, and product manuals, instructions, promotional and marketing materials, and/or technical materials to distributors, resellers, and end users—encouraging other's infringement of the '978 patent with the specific intent to induce such infringement. Users of the '978 Accused Products infringe through the

ordinary use of such products. Google further induces others to infringe by marketing the accused products VR capabilities. Google further induces others to infringe by marketing the accused products VR and/or AR capabilities. *See* https://store.google.com/us/product/pixel_2?utm_source=en-ha-na-sem&utm_medium=text&utm_term=USsale&ds_kid=43700024791423013&utm_content=bkws&utm_campaign=Pixel&gclid=Cj0KCQiAyszSBRDJARIsAHAqQ4ouuVPL5FfpJ6LKd-eFT_H963mAond-6M5FDxrDHnEZrtiKvPQsGFwaAqRGEALw_wcB&gclsrc=aw.ds&dclid=CPCzr7D8yNgCFYYC0wodJwwB-Q ("Better with Pixel High quality VR, wherever you go."); *See also* https://store.google.com/us/product/google_daydream_view (describing "Google Daydream View."). CyWee expects discovery to provide more facts relevant to Google's induced infringement.

125. The Google Pixel includes a 3-axis geomagnetic sensor.

126. The Google Pixel includes a 3-axis geomagnetic sensor that is capable of measuring a geomagnetic field.

127. The Google Pixel includes a 3-axis geomagnetic field sensor to measure a geomagnetic field using a "Sensor Coordinate System." *See* https://developer.android.com/guide/topics/sensors/sensors_overview.html (describing "Sensor Coordinate System").

128. The Android operating system that runs on the Google Pixel uses the measurement from a 3-axis geomagnetic sensor included in the device.

129. The Android operating system that runs on the Google Pixel uses the measurement from a 3-axis accelerometer, the measurement from a 3-axis geomagnetic field sensor, and the measurement from a 3-axis gyroscope to calculate an attitude of the device.

130. The Android operating system that runs on the Google Pixel uses the measurement from a 3-axis accelerometer, the measurement from a 3-axis geomagnetic field sensor, and the measurement from a 3-axis gyroscope to calculate an attitude of the device that can be represented by an azimuth angle, a pitch angle, and a roll angle.

131. The Google Pixel has the ability to directly control apps by moving or rotating the device (for example, racing game apps).

132. The Google Pixel has the ability to run apps that can provide information based on the direction your device is facing, such as a map or navigation app.

133. The Google Pixel 2 includes a 3-axis geomagnetic sensor.

134. The Google Pixel 2 includes a 3-axis geomagnetic sensor that is capable of measuring a geomagnetic field.

135. The Google Pixel 2 includes a 3-axis geomagnetic field sensor to measure a geomagnetic field using a “Sensor Coordinate System.” *See*

https://developer.android.com/guide/topics/sensors/sensors_overview.html (describing "Sensor Coordinate System").

136. The Android operating system that runs on the Google Pixel 2 uses the measurement from a 3-axis geomagnetic sensor included in the device.

137. The Android operating system that runs on the Google Pixel 2 uses the measurement from a 3-axis accelerometer, the measurement from a 3-axis geomagnetic field sensor, and the measurement from a 3-axis gyroscope to calculate an attitude of the device.

138. The Android operating system that runs on the Google Pixel 2 uses the measurement from a 3-axis accelerometer, the measurement from a 3-axis geomagnetic field sensor, and the measurement from a 3-axis gyroscope to calculate an attitude of the device that can be represented by an azimuth angle, a pitch angle, and a roll angle.

139. The Google Pixel 2 has the ability to directly control apps by moving or rotating the device (for example, racing game apps).

140. The Google Pixel 2 has the ability to run apps that can provide information based on the direction your device is facing, such as a map or navigation app. The Google Pixel includes a 3-axis geomagnetic sensor.

141. The Google Pixel XL includes a 3-axis geomagnetic sensor that is capable of measuring a geomagnetic field.

142. The Google Pixel XL includes a 3-axis geomagnetic field sensor to measure a geomagnetic field using a "Sensor Coordinate System." *See* https://developer.android.com/guide/topics/sensors/sensors_overview.html (describing "Sensor Coordinate System").

143. The Android operating system that runs on the Google Pixel XL uses the measurement from a 3-axis geomagnetic sensor included in the device.

144. The Android operating system that runs on the Google Pixel XL uses the measurement from a 3-axis accelerometer, the measurement from a 3-axis geomagnetic field sensor, and the measurement from a 3-axis gyroscope to calculate an attitude of the device.

145. The Android operating system that runs on the Google Pixel XL uses the measurement from a 3-axis accelerometer, the measurement from a 3-axis geomagnetic field sensor, and the measurement from a 3-axis gyroscope to calculate an attitude of the device that can be represented by an azimuth angle, a pitch angle, and a roll angle.

146. The Google Pixel XL has the ability to directly control apps by moving or rotating the device (for example, racing game apps).

147. The Google Pixel XL has the ability to run apps that can provide information based on the direction your device is facing, such as a map or navigation app.

148. The Google Pixel 2 XL includes a 3-axis geomagnetic sensor.

149. The Google Pixel 2 XL includes a 3-axis geomagnetic sensor that is capable of measuring a geomagnetic field.

150. The Google Pixel 2 XL includes a 3-axis geomagnetic field sensor to measure a geomagnetic field using a "Sensor Coordinate System." *See* https://developer.android.com/guide/topics/sensors/sensors_overview.html (describing "Sensor Coordinate System").

151. The Android operating system that runs on the Google Pixel 2 XL uses the measurement from a 3-axis geomagnetic sensor included in the device.

152. The Android operating system that runs on the Google Pixel 2 XL uses the measurement from a 3-axis accelerometer, the measurement from a 3-axis geomagnetic field sensor, and the measurement from a 3-axis gyroscope to calculate an attitude of the device.

153. The Android operating system that runs on the Google Pixel 2 XL uses the measurement from a 3-axis accelerometer, the measurement from a 3-axis geomagnetic field sensor, and the measurement from a 3-axis gyroscope to calculate an attitude of the device that can be represented by an azimuth angle, a pitch angle, and a roll angle.

154. The Google Pixel 2 XL has the ability to directly control apps by moving or rotating the device (for example, racing game apps).

155. The Google Pixel 2 XL has the ability to run apps that can provide information based on the direction your device is facing, such as a map or navigation app.

156. Google's actions with regards to distributors, resellers, and/or end users of the '978 Accused Products induce infringement of the patent by others, and at least as a result of the filing of this Complaint, Google is aware that its actions induce infringement. Despite such awareness, upon information and belief, Google will continue to take active steps—such as creating and disseminating the '978 Accused Products, accessories thereto, product manuals, instructions, support materials, promotional and marketing materials, and/or technical materials to distributors, resellers, and end users—encouraging others to infringe the '978 Patent with the specific intent to induce such infringement.

157. Google provides manuals and/or instructions for the '978 Accused Products and/or provides instructional and support materials on its website that teach and instruct its customers to operate those products in ways that practice the claimed invention. For instance, Google provides support documents for the '978 Accused Products that instruct users on how to use motion gestures to perform various device functions. *See, e.g.*, https://support.google.com/pixelphone/answer/7 443425?hl=en (last visited April 4, 2018) (describing how to check device notifications by lifting a '978 Accused

Product and how to switch between front and rear facing cameras using a "double-twist gesture").

158. Additionally, Google offers the Google Daydream View as an accessory for use with the '978 Accused Products and other products that practice the '978 patent and specifically advertises and instructs users in how to use a '978 Accused Product with the Google Daydream View in a manner that infringes the '978 patent. *See* https://store.google.com/product/google_daydream_view (last visited April 4, 2018); *see also* https://store.google.com/product/google_daydream_ view_learn (last visited April 4, 2018) (describing the "Swivel and Play" feature of the Google Daydream View and instructing users to "[u]se a swivel chair to spin 360° and see all around you.").

159. These support materials and accessories, among others, demonstrate that Google actively induces users to operate their products in ways that practice the claimed invention. Upon information and belief, Google will continue to instruct end-users of the '978 Accused Products to operate those products in ways that practice the claimed invention even after being put on actual notice of the infringement of the '978 Patent.

160. CyWee adopts, and incorporates by reference, as if fully stated herein, the attached claim chart for claim 10 of the '978 patent, which is attached hereto as **Exhibit E**. The claim chart describes and demonstrates how Google infringes the

’978 patent. In addition, CyWee alleges that Google infringes one or more additional claims of the ’978 patent in a similar manner.

161. Google’s acts of infringement have caused and will continue to cause substantial and irreparable damage to CyWee.

162. As a result of Google’s infringement of the ’978 patent, CyWee has been damaged. CyWee is, therefore, entitled to damages pursuant to 35 U.S.C. § 284 in an amount that presently cannot be pled but that will be determined at trial.

PRAYER FOR RELIEF

WHEREFORE, PREMISES CONSIDERED, Plaintiff prays for entry of judgment against Google as follows:

A. A judgment that Google has infringed and continue to infringe the ’438 patent and ’978 patent, directly and/or indirectly, as alleged herein;

B. That Google provide to CyWee an accounting of all gains, profits, and advantages derived by Google’s infringement of the ’438 patent and ’978 patent, and that CyWee be awarded damages adequate to compensate them for the wrongful infringement by Googles, in accordance with 35 U.S.C. § 284;

C. That CyWee be awarded any other supplemental damages and interest on all damages, including, but not limited to, attorneys' fees available under 35 U.S.C. § 285;

D. That the Court permanently enjoin Google and all those in privity with Googles from making, having made, selling, offering for sale, distributing, and/or using products that infringe the '438 patent and '978 patent, including the '438 Accused Products and/or '978 Accused Products, in the United States; and

E. That CyWee be awarded such other and further relief and all remedies available at law.

DEMAND FOR JURY TRIAL

Pursuant to Federal Rule of Civil Procedure 38(b), CyWee hereby demands a trial by jury on all issues triable to a jury.

Dated: April 16, 2018

Respectfully submitted,

/s/ *Stamatios Stamoulis*

STAMOULIS & WEINBLATT LLC
Stamatios Stamoulis (#4606)
stamoulis@swdelaw.com
Richard C. Weinblatt (#5080)
weinblatt@swdelaw.com
Two Fox Point Centre
6 Denny Road, Suite 307
Wilmington, Delaware 19809
Telephone: (302) 999-1540

Michael W. Shore (Texas 18294915)
mshore@shorechan.com
Alfonso G. Chan (Texas 24012408)
achan@shorechan.com
Christopher Evans (Texas 24058901)
cevans@shorechan.com
Ari B. Rafilson (Texas 24060456)
arafilson@shorechan.com
William D. Ellerman (Texas 24007151)
wellerman@shorechan.com
Paul T. Beeler (Texas 24095432)
pbeeler@shorechan.com)
SHORE CHAN DEPUMPO LLP
901 Main Street, Suite 3300
Dallas, Texas 75202
Telephone (214) 593-9110
Facsimile (214) 593-9111

COUNSEL FOR PLAINTIFF
CYWEE GROUP, LTD.

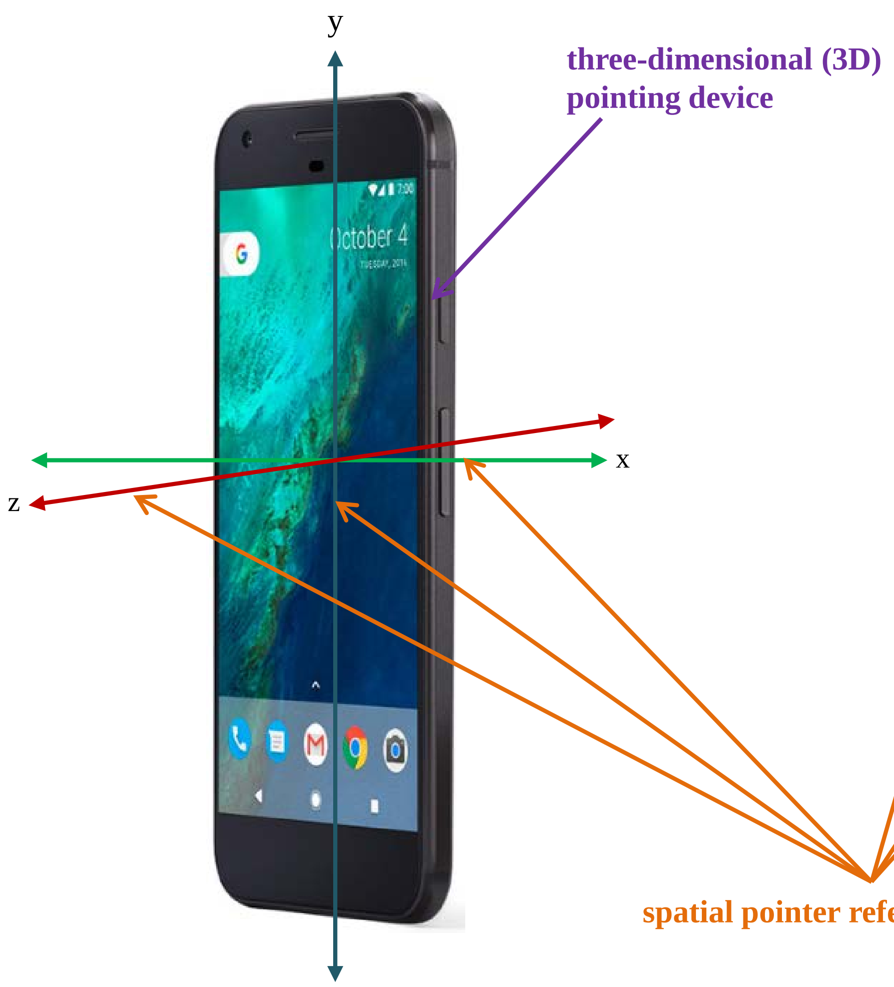
EXHIBIT D

U.S. Patent No. 8,441,438

Google Pixel

Claim 14

A method for obtaining a resulting deviation including resultant angles in a spatial pointer reference frame of a three-dimensional (3D) pointing device utilizing a six-axis motion sensor module therein and subject to movements and rotations in dynamic environments in said spatial pointer reference frame, comprising the steps of:



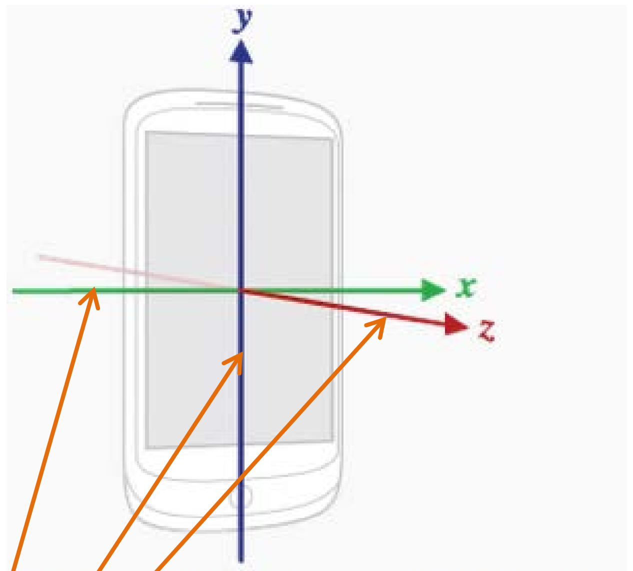


Figure 1. Coordinate system (relative to a device) that's used by the Sensor API.

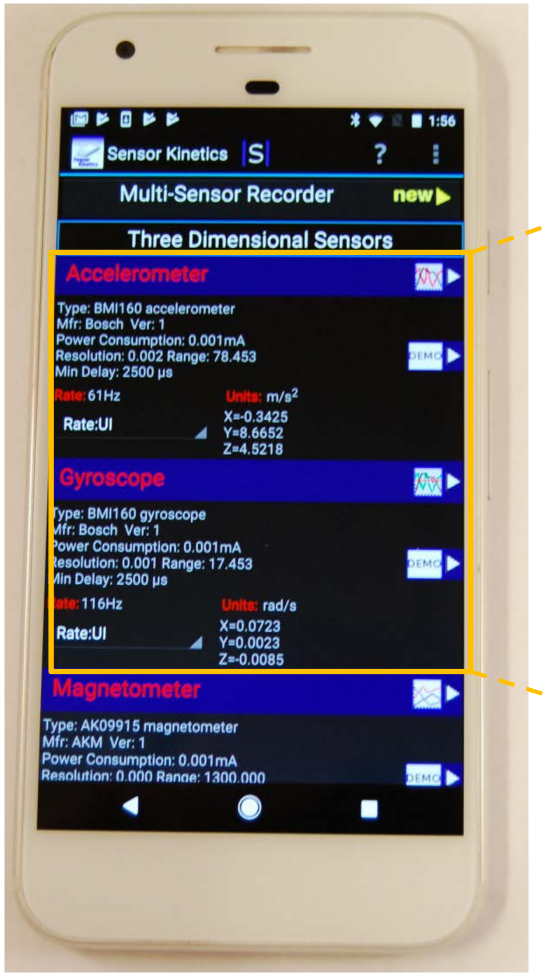
spatial pointer reference frame

Source: http://developer.android.com/guide/topics/sensors/sensors_overview.html#sensors-coords

Claim 14

A method for obtaining a resulting deviation including resultant angles in a spatial pointer reference frame of a three-dimensional (3D) pointing device utilizing **a six-axis motion sensor module** therein and subject to movements and rotations in dynamic environments in said spatial pointer reference frame, comprising the steps of:

The **six-axis motion sensor module** includes the accelerometer and gyroscope.

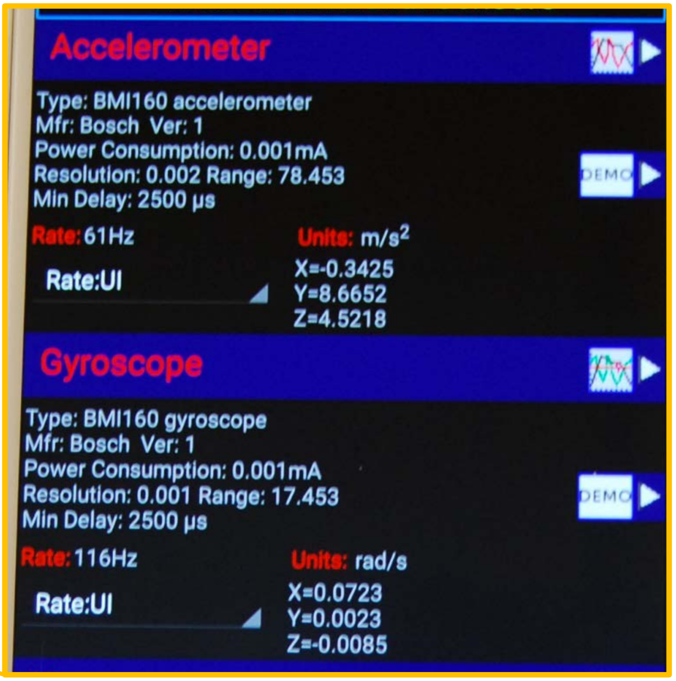


1:56
Sensor Kinetics
Multi-Sensor Recorder new
Three Dimensional Sensors

Accelerometer
Type: BMI160 accelerometer
Mfr: Bosch Ver: 1
Power Consumption: 0.001mA
Resolution: 0.002 Range: 78.453
Min Delay: 2500 µs
Rate: 61Hz
Rate:UI
Units: m/s²
X=-0.3425
Y=8.6652
Z=4.5218

Gyroscope
Type: BMI160 gyroscope
Mfr: Bosch Ver: 1
Power Consumption: 0.001mA
Resolution: 0.001 Range: 17.453
Min Delay: 2500 µs
Rate: 116Hz
Rate:UI
Units: rad/s
X=0.0723
Y=0.0023
Z=-0.0085

Magnetometer
Type: AK09915 magnetometer
Mfr: AKM Ver: 1
Power Consumption: 0.001mA
Resolution: 0.000 Range: 1300.000



Accelerometer
Type: BMI160 accelerometer
Mfr: Bosch Ver: 1
Power Consumption: 0.001mA
Resolution: 0.002 Range: 78.453
Min Delay: 2500 µs
Rate: 61Hz
Rate:UI
Units: m/s²
X=-0.3425
Y=8.6652
Z=4.5218

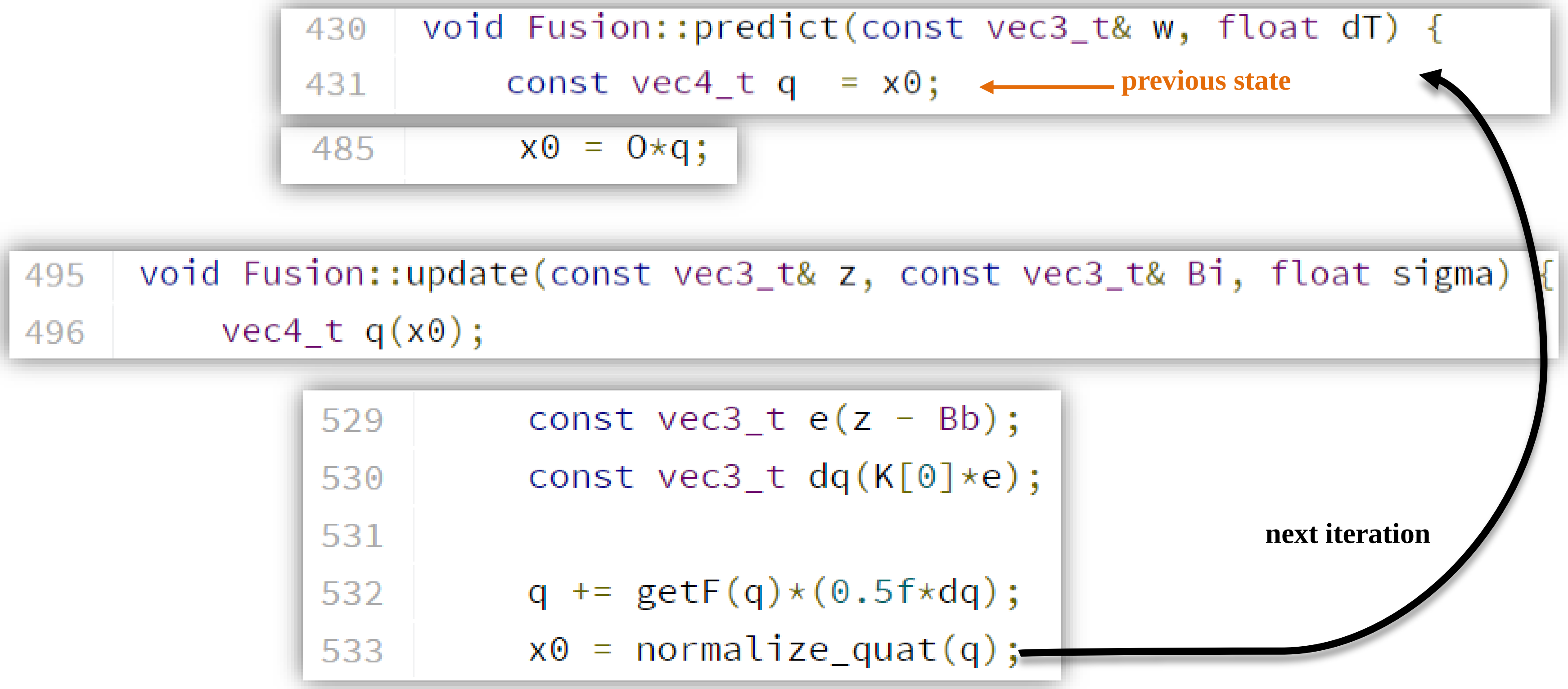
Gyroscope
Type: BMI160 gyroscope
Mfr: Bosch Ver: 1
Power Consumption: 0.001mA
Resolution: 0.001 Range: 17.453
Min Delay: 2500 µs
Rate: 116Hz
Rate:UI
Units: rad/s
X=0.0723
Y=0.0023
Z=-0.0085

SUBJECT TO CHANGE

Claim 14

obtaining a **previous state** of the six-axis motion sensor module; wherein the **previous state** includes an initial-value set associated with **previous angular velocities** gained from the motion sensor signals of the six-axis motion sensor module at a previous time T−1;

The previous state is obtained through an update program that includes a `predict()` function and an `update()` function. Those functions that are used to update the global variable `x0` based on `x0` (the **previous state**) associated with **previous angular velocities** `w` gained at a previous time T-1 to obtain an updated state `x0`. The updated state `x0` becomes the previous state `x0` at time T (the next iteration) of the update program to obtain the updated state `x0` at time T.



```
void Fusion::predict(const vec3_t& w, float dT) {
    const vec4_t q  = x0;
```

```
        x0 = O*q;
```

```
void Fusion::update(const vec3_t& z, const vec3_t& Bi, float sigma) {
    vec4_t q(x0);
```

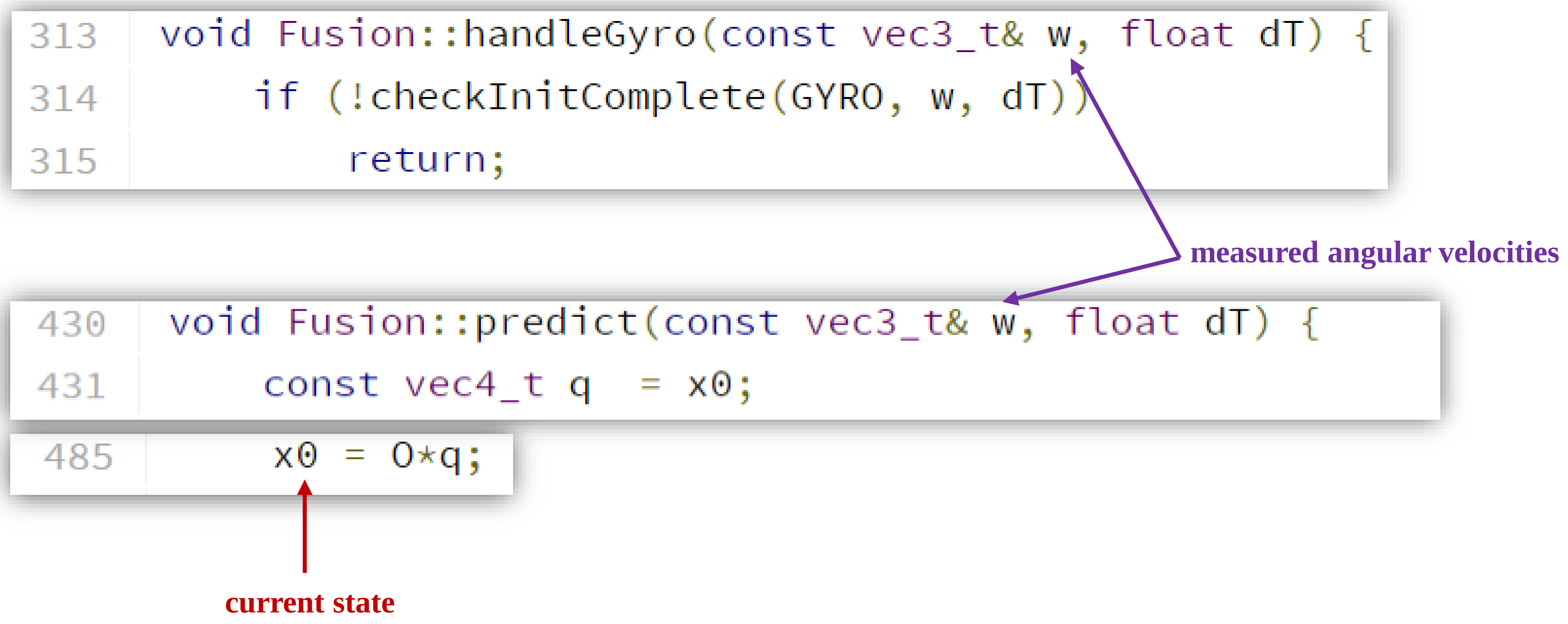
```
    const vec3_t e(z - Bb);
    const vec3_t dq(K[0]*e);

    q += getF(q)*(0.5f*dq);
    x0 = normalize_quat(q);
```

Source: https://android.googlesource.com/platform/frameworks/native/+/master/services/sensorservice/Fusion.cpp

Claim 14

obtaining a **current state** of the six-axis motion sensor module by obtaining **measured angular velocities** ω_x, ω_y, ω_z gained from the motion sensor signals of the six-axis motion sensor module at a current time T;

The `predict()` function runs during each iteration of the fusion algorithm, at a time T its output represents a **current state** output as `x0`. The `predict()` function is called by the `handleGyro()` function and receives **measured angular velocities**, `w`, associated with the **current state**.



```
void Fusion::handleGyro(const vec3_t& w, float dT) {
    if (!checkInitComplete(GYRO, w, dT))
        return;
```

```
void Fusion::predict(const vec3_t& w, float dT) {
    const vec4_t q  = x0;
```

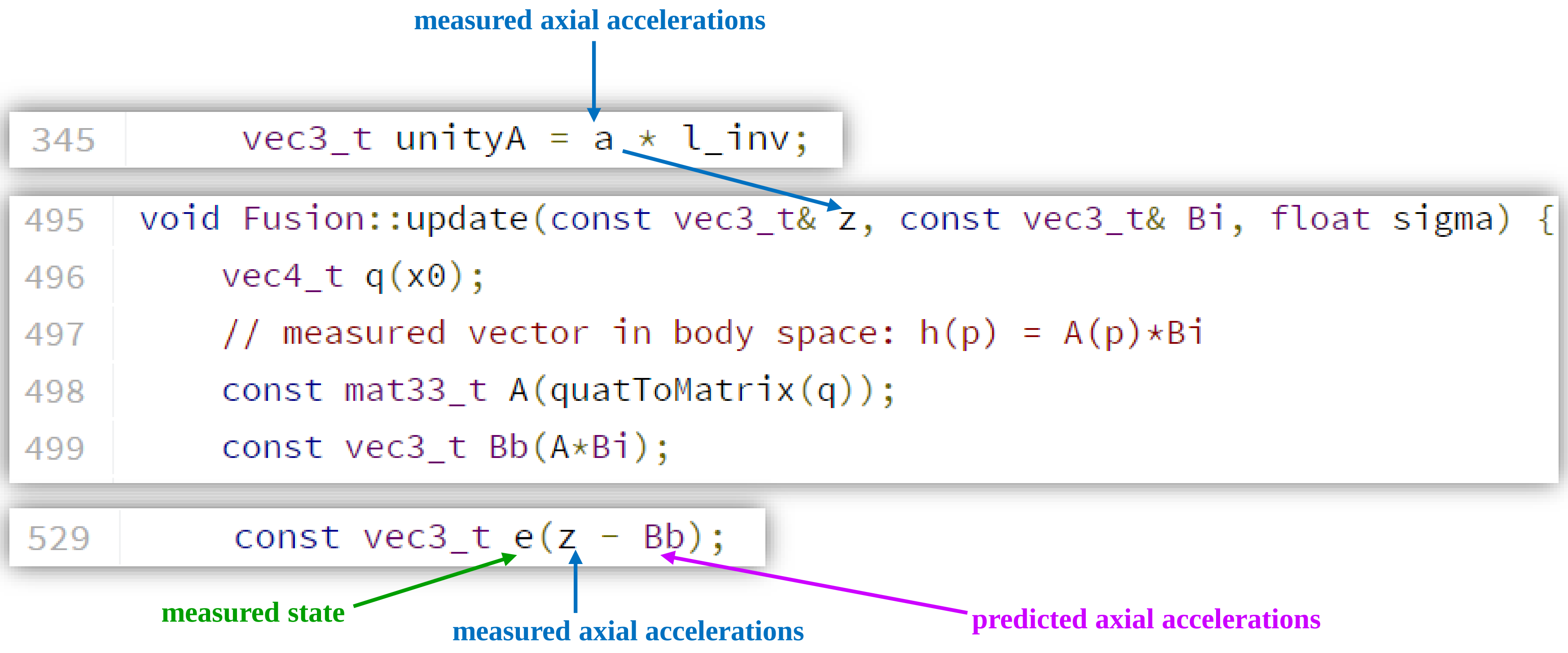
```
        x0 = O*q;
```

Source: https://android.googlesource.com/platform/frameworks/native/+/master/services/sensorservice/Fusion.cpp

Claim 14

obtaining a **measured state** of the six-axis motion sensor module by obtaining **measured axial accelerations** Ax, Ay, Az gained from the motion sensor signals of the six-axis motion sensor module at the current time T and calculating **predicted axial accelerations** Ax′, Ay′, Az′ based on the **measured angular velocities** ω_x, ω_y, ω_z of the current state of the six-axis motion sensor module **without using any derivatives of the measured angular velocities** ωx, ωy, ωz;

The variable e is a **measured state** that includes **measured axil accelerations** z and **predicted axial accelerations** Bb calculated based on x0 (the previous state, which is calculated based on the **measured angular velocities**).


measured axial accelerations
measured state
measured axial accelerations
predicted axial accelerations

```
        vec3_t unityA = a * l_inv;
```

```
void Fusion::update(const vec3_t& z, const vec3_t& Bi, float sigma) {
    vec4_t q(x0);
    // measured vector in body space: h(p) = A(p)*Bi
    const mat33_t A(quatToMatrix(q));
    const vec3_t Bb(A*Bi);
```

```
        const vec3_t e(z - Bb);
```

As shown in the code above, the predicted measurement is obtained based on the first signal set **without using any derivatives of the measured angular velocities**.

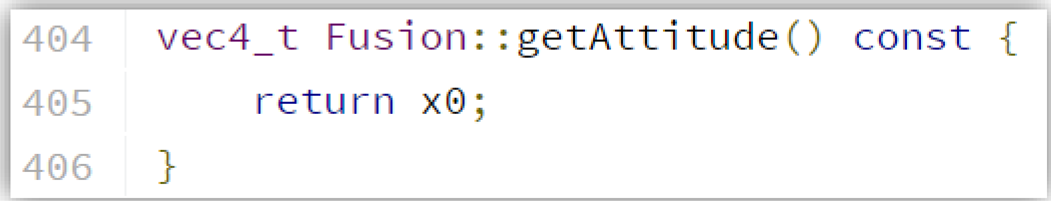
Source: https://android.googlesource.com/platform/frameworks/native/+/master/services/sensorservice/Fusion.cpp

SUBJECT TO CHANGE

Claim 14

said **current state** of the six-axis motion sensor module is a second quaternion with respect to said current time T;

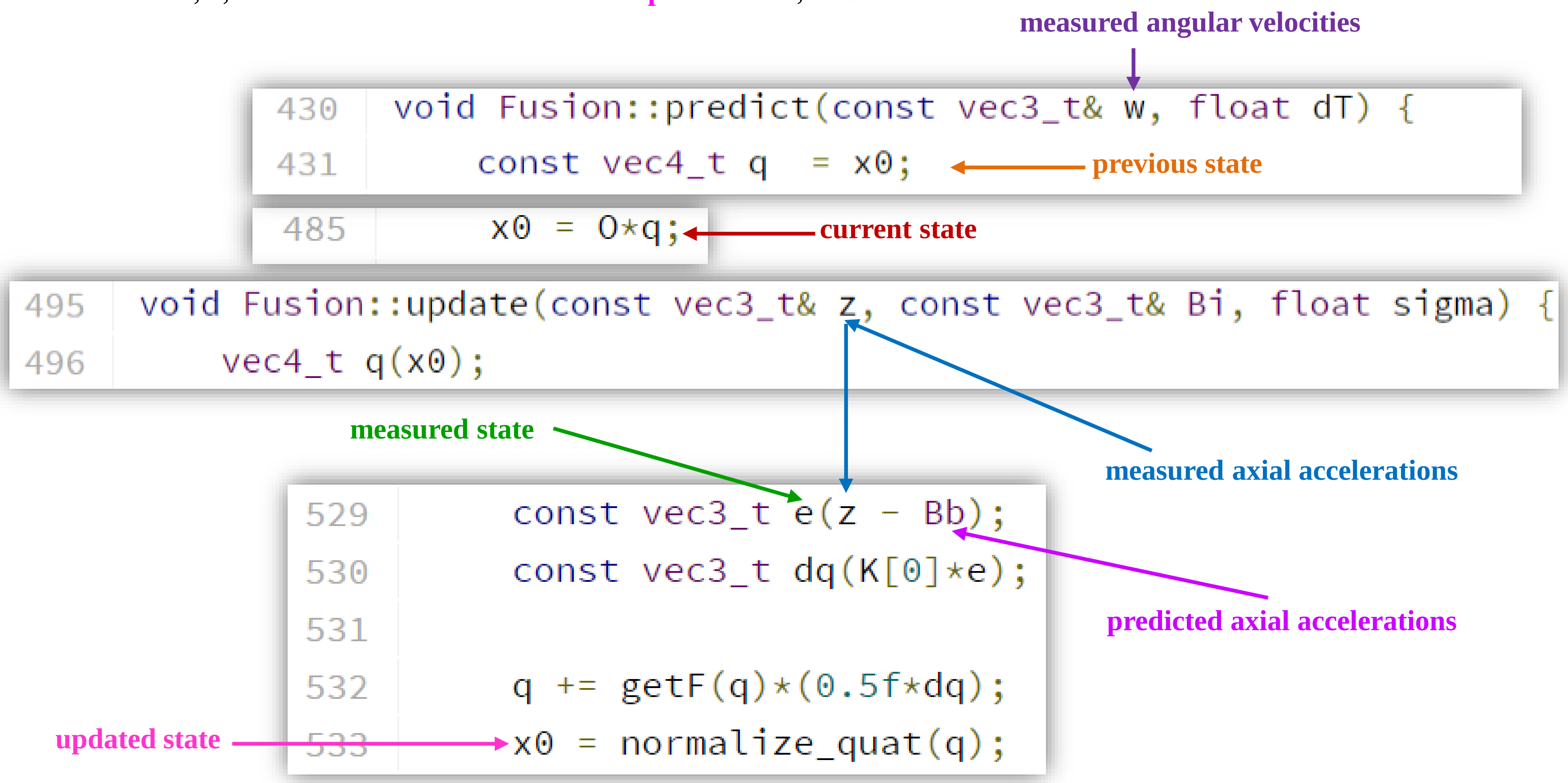
As shown in the examples provided, the **current state** is represented by the global state variable x0, which is a quaternion with respect to the current time T.



```
vec4_t Fusion::getAttitude() const {
    return x0;
}
```

Source: https://android.googlesource.com/platform/frameworks/native/+/master/services/sensorservice/Fusion.cpp

Claim 14
comparing the second quaternion in relation to the **measured angular velocities** ω_x, ω_y, ω_z of the **current state** at current time T with the **measured axial accelerations** Ax, Ay, Az and the **predicted axial accelerations** Ax′, Ay′, Az′ also at current time T; obtaining an **updated state** of the six-axis motion sensor module by comparing the **current state** with the **measured state** of the six-axis motion sensor module; and
For example, as previously shown, the **measured state**, `e`, is obtained using the `update()` function, which combines the **measured axial accelerations**, `z`, and the **predicted axial accelerations**, `Bb`. Moreover, the **predicted axial accelerations** are determined based on the **measured angular velocities** of the **current state** at the current time T. The update() function further compares the **measured state**, `e`, and the **current state** to obtain the **updated state**, `x0`.



```
void Fusion::predict(const vec3_t& w, float dT) {
    const vec4_t q  = x0;
```

```
    x0 = O*q;
```

```
void Fusion::update(const vec3_t& z, const vec3_t& Bi, float sigma) {
    vec4_t q(x0);
```

```
    const vec3_t e(z - Bb);
    const vec3_t dq(K[0]*e);

    q += getF(q)*(0.5f*dq);
    x0 = normalize_quat(q);
```

measured angular velocities
previous state
current state
measured state
measured axial accelerations
predicted axial accelerations
updated state

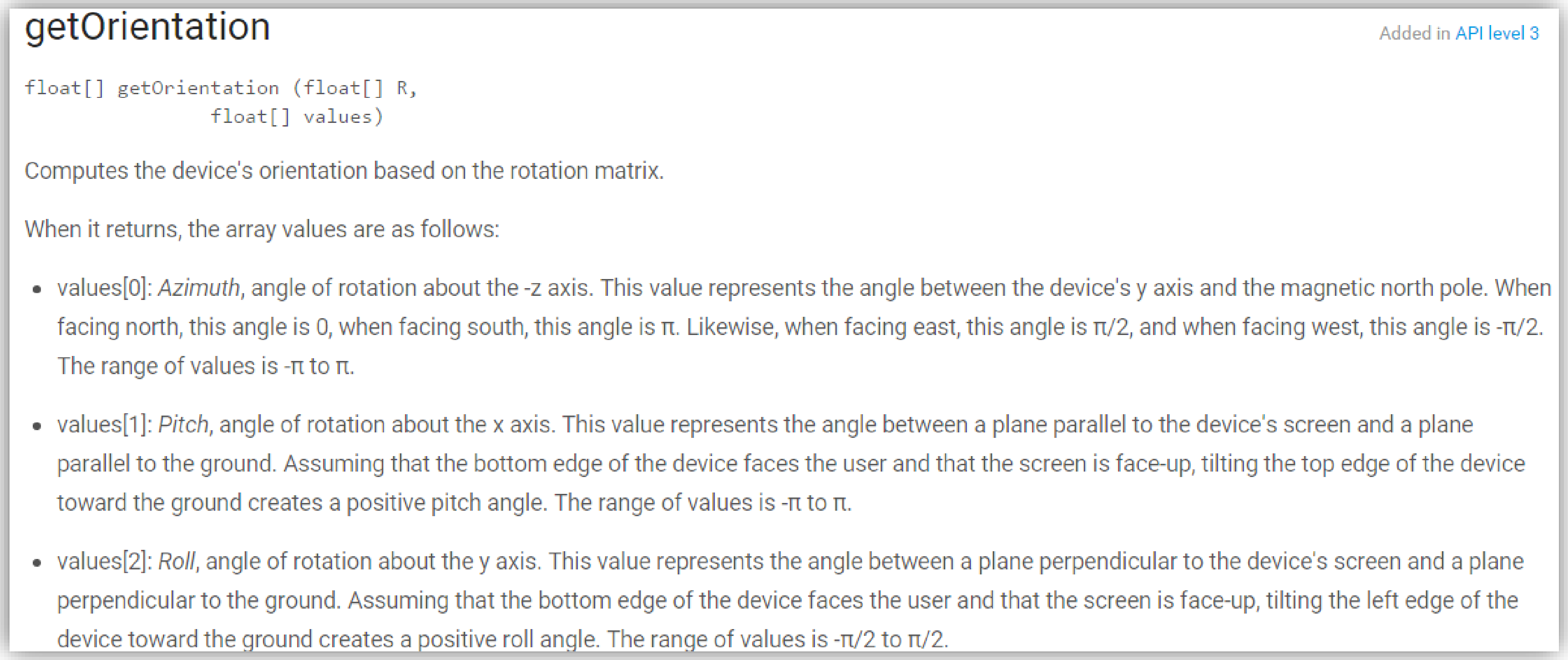
Source: https://android.googlesource.com/platform/frameworks/native/+/master/services/sensorservice/Fusion.cpp

Claim 14

calculating and converting the **updated state** of the six axis motion sensor module to said **resulting deviation comprising said resultant angles** in said spatial pointer reference frame of the 3D pointing device.

The **updated state** x0 is in quaternion form, and can easily be converted to resultant angles.

According to Android's developer library, the getOrientation() function "computes the device's orientation based on the rotation matrix," and returns **resultant angles** including the Azimuth, Pitch, and Roll angles.



getOrientation

Added in API level 3

```
float[] getOrientation (float[] R,
                float[] values)
```

Computes the device's orientation based on the rotation matrix.

When it returns, the array values are as follows:

- values[0]: *Azimuth*, angle of rotation about the -z axis. This value represents the angle between the device's y axis and the magnetic north pole. When facing north, this angle is 0, when facing south, this angle is π. Likewise, when facing east, this angle is π/2, and when facing west, this angle is -π/2. The range of values is -π to π.
- values[1]: *Pitch*, angle of rotation about the x axis. This value represents the angle between a plane parallel to the device's screen and a plane parallel to the ground. Assuming that the bottom edge of the device faces the user and that the screen is face-up, tilting the top edge of the device toward the ground creates a positive pitch angle. The range of values is -π to π.
- values[2]: *Roll*, angle of rotation about the y axis. This value represents the angle between a plane perpendicular to the device's screen and a plane perpendicular to the ground. Assuming that the bottom edge of the device faces the user and that the screen is face-up, tilting the left edge of the device toward the ground creates a positive roll angle. The range of values is -π/2 to π/2.

The getRotationMatrixFromVector() function "convert[s] a rotation vector to a rotation matrix," and the getQuaternionFromVector() function "convert[s] a rotation vector to a normalized quaternion." Therefore, the quaternion, x0, can be easily converted to its mathematically equivalent form, rotation matrix, and used by getOrientation() function to compute the orientation in its angular form.

Source: https://developer.android.com/reference/android/hardware/SensorManager.html#getOrientation(float[],%20float[])

SUBJECT TO CHANGE

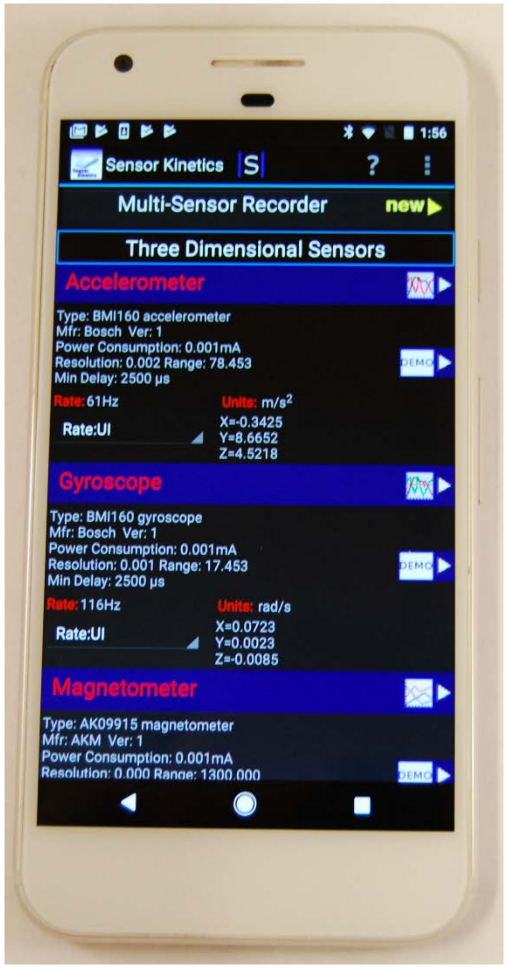
EXHIBIT E

U.S. Patent No. 8,552,978

Google Pixel

Claim 10

A method for compensating rotations of a 3D pointing device, comprising:


1:56

Sensor Kinetics S ? ⋮

Multi-Sensor Recorder new▶

Three Dimensional Sensors

Accelerometer

Type: BMI160 accelerometer
Mfr: Bosch Ver: 1
Power Consumption: 0.001mA
Resolution: 0.002 Range: 78.453
Min Delay: 2500 μs

Rate: 61Hz
Rate:UI
Units: m/s^2
X=-0.3425
Y=8.6652
Z=4.5218

Gyroscope

Type: BMI160 gyroscope
Mfr: Bosch Ver: 1
Power Consumption: 0.001mA
Resolution: 0.001 Range: 17.453
Min Delay: 2500 μs

Rate: 116Hz
Rate:UI
Units: rad/s
X=0.0723
Y=0.0023
Z=-0.0085

Magnetometer

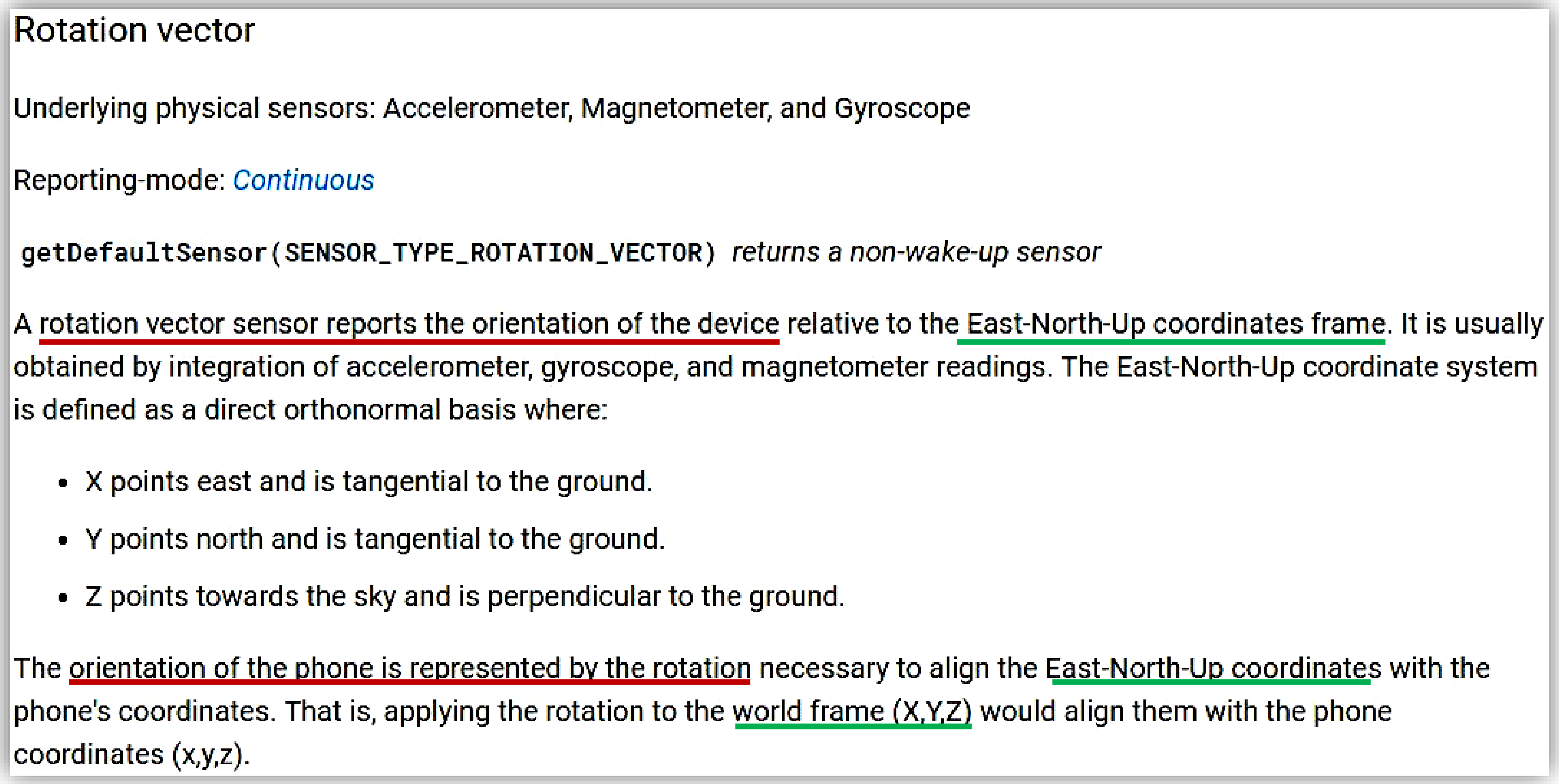
Type: AK09915 magnetometer
Mfr: AKM Ver: 1
Power Consumption: 0.001mA
Resolution: 0.000 Range: 1300.000

Google Pixel

Claim 10

generating an **orientation output** associated with an orientation of the 3D pointing device associated with three coordinate axes of a **global reference frame associated with Earth**;

When the orientation sensor is software-based, the **orientation output** is the attitude of the device that can be represented by the azimuth, pitch, and roll angles relative to the magnetic North Pole associated with a **global reference frame associated with Earth**.



Rotation vector

Underlying physical sensors: Accelerometer, Magnetometer, and Gyroscope

Reporting-mode: *Continuous*

`getDefaultSensor(SENSOR_TYPE_ROTATION_VECTOR)` *returns a non-wake-up sensor*

A rotation vector sensor reports the orientation of the device relative to the East-North-Up coordinates frame. It is usually obtained by integration of accelerometer, gyroscope, and magnetometer readings. The East-North-Up coordinate system is defined as a direct orthonormal basis where:

- X points east and is tangential to the ground.
- Y points north and is tangential to the ground.
- Z points towards the sky and is perpendicular to the ground.

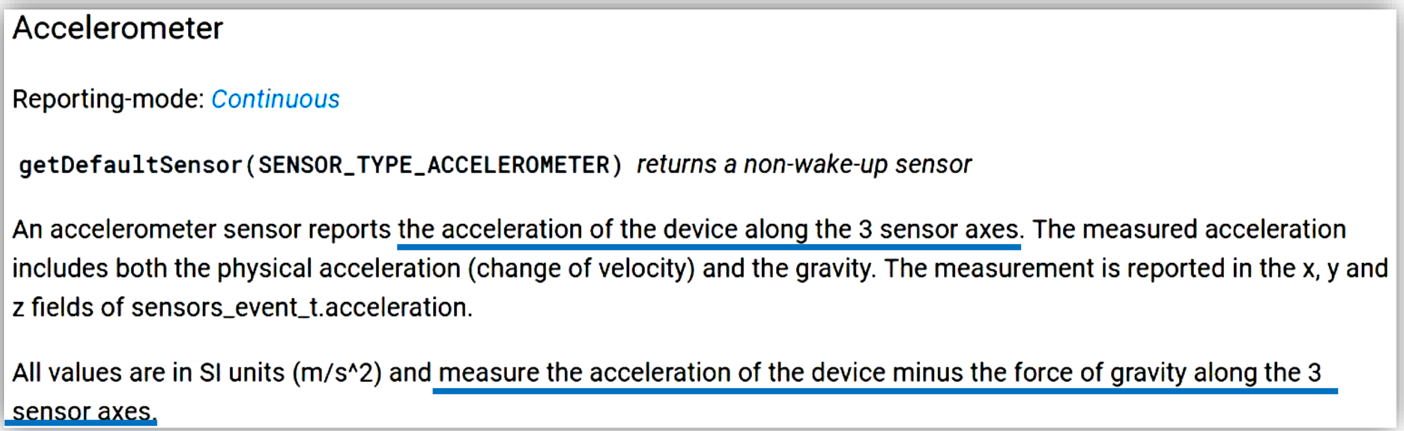
The orientation of the phone is represented by the rotation necessary to align the East-North-Up coordinates with the phone's coordinates. That is, applying the rotation to the world frame (X,Y,Z) would align them with the phone coordinates (x,y,z).

Source: https://source.android.com/devices/sensors/sensor-types#rotation_vector

SUBJECT TO CHANGE

Claim 10

generating a **first signal set** comprising axial accelerations associated with movements and rotations of the 3D pointing device in the **spatial reference frame**;



Accelerometer

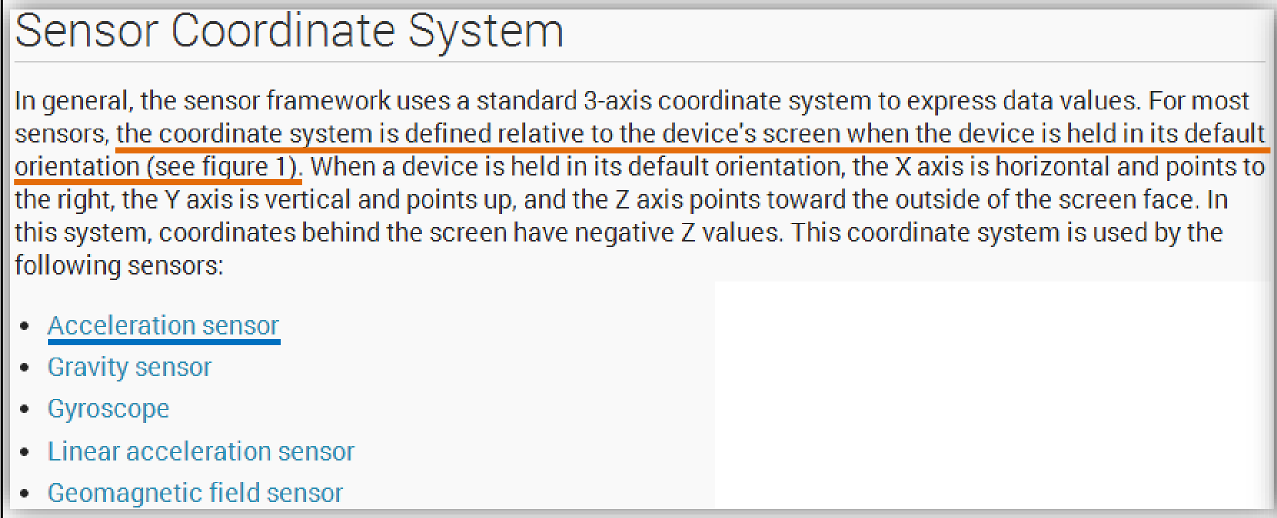
Reporting-mode: *Continuous*

`getDefaultSensor(SENSOR_TYPE_ACCELEROMETER)` *returns a non-wake-up sensor*

An accelerometer sensor reports the acceleration of the device along the 3 sensor axes. The measured acceleration includes both the physical acceleration (change of velocity) and the gravity. The measurement is reported in the x, y and z fields of sensors_event_t.acceleration.

All values are in SI units (m/s^2) and measure the acceleration of the device minus the force of gravity along the 3 sensor axes.

Source: https://source.android.com/devices/sensors/sensor-types#accelerometer



Sensor Coordinate System

In general, the sensor framework uses a standard 3-axis coordinate system to express data values. For most sensors, the coordinate system is defined relative to the device's screen when the device is held in its default orientation (see figure 1). When a device is held in its default orientation, the X axis is horizontal and points to the right, the Y axis is vertical and points up, and the Z axis points toward the outside of the screen face. In this system, coordinates behind the screen have negative Z values. This coordinate system is used by the following sensors:

- Acceleration sensor
- Gravity sensor
- Gyroscope
- Linear acceleration sensor
- Geomagnetic field sensor

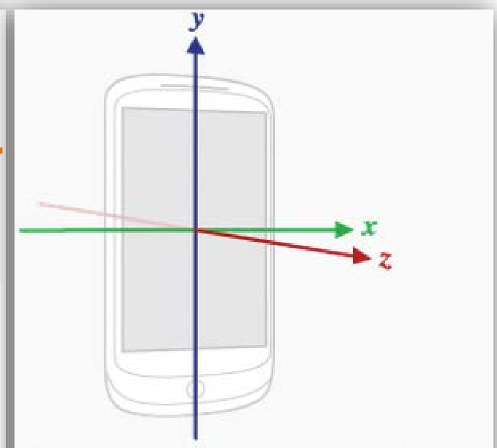
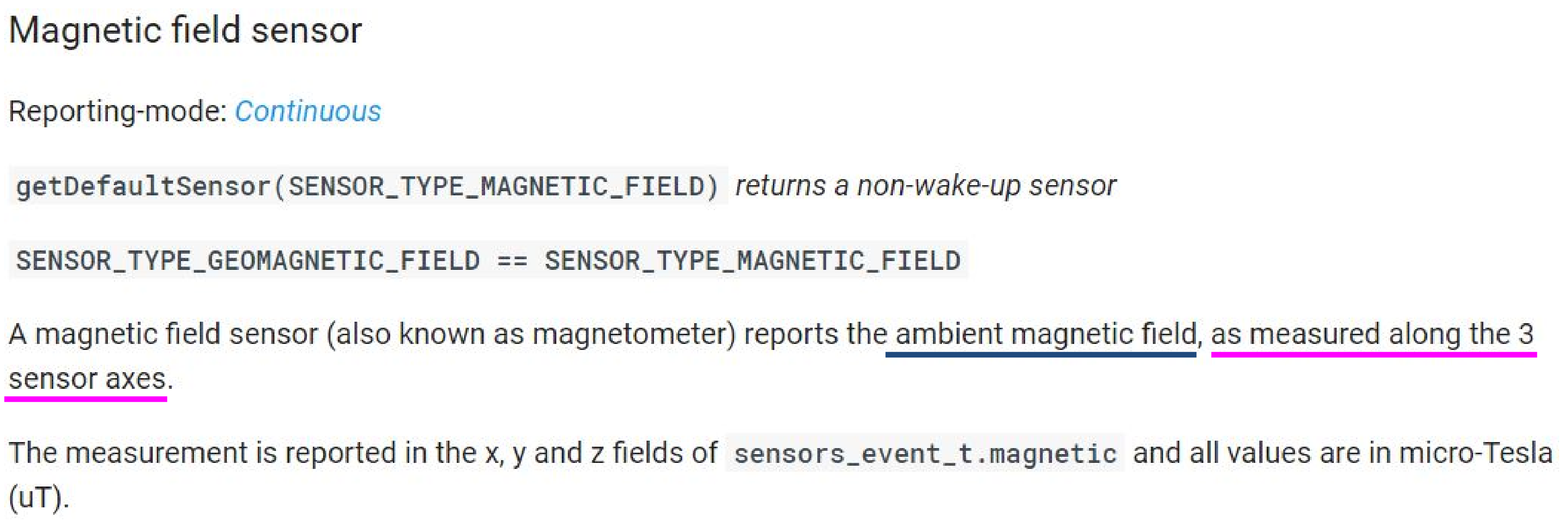


Figure 1. Coordinate system (relative to a device) that's used by the Sensor API.

Source: http://developer.android.com/guide/topics/sensors/sensors_overview.html#sensors-coords

Claim 10
generating a **second signal set** associated with **Earth's magnetism**;
The magnetometer (i.e., the compass) generates a second signal set associated with **Earth's magnetism**.



Magnetic field sensor

Reporting-mode: *Continuous*

`getDefaultSensor(SENSOR_TYPE_MAGNETIC_FIELD)` *returns a non-wake-up sensor*

`SENSOR_TYPE_GEOMAGNETIC_FIELD == SENSOR_TYPE_MAGNETIC_FIELD`

A magnetic field sensor (also known as magnetometer) reports the ambient magnetic field, as measured along the 3 sensor axes.

The measurement is reported in the x, y and z fields of `sensors_event_t.magnetic` and all values are in micro-Tesla (uT).

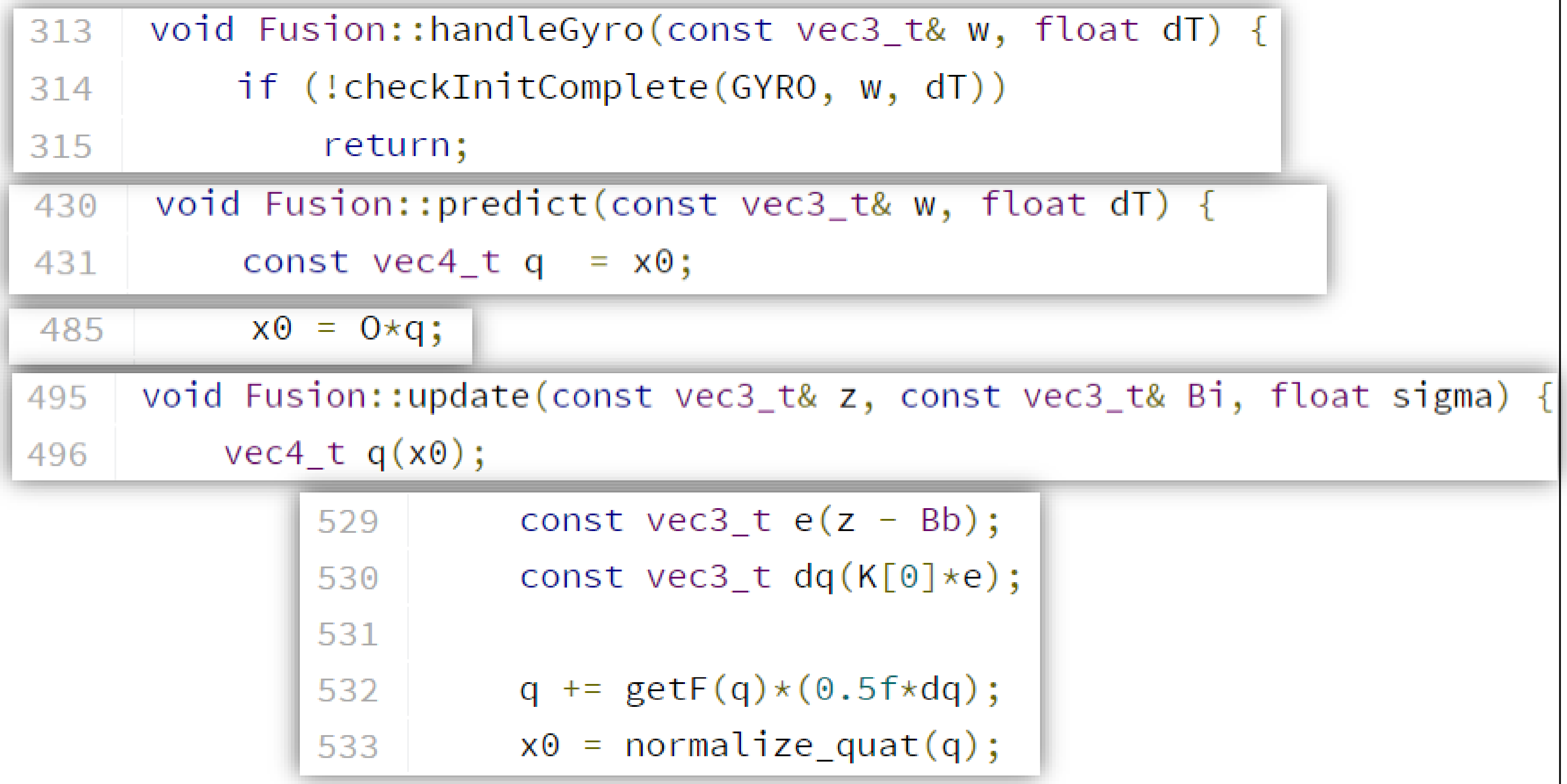
Source: https://source.android.com/devices/sensors/sensor-types#magnetic_field_sensor

Claim 10

generating the **orientation output** based on the **first signal set**, the **second signal set** and the **rotation output** or based on the **first signal set** and the **second signal set**;

The Android source code shows generating the **orientation output** based on the **first signal set**, the **second signal set** and the **rotation output**.

The `handleGyro()` function passes **rotation output** `w` to the `predict()` function and the `update()` function to calculate an **orientation output**, `x0`.



```
void Fusion::handleGyro(const vec3_t& w, float dT) {
    if (!checkInitComplete(GYRO, w, dT))
        return;
```

```
void Fusion::predict(const vec3_t& w, float dT) {
    const vec4_t q  = x0;
```

```
    x0 = O*q;
```

```
void Fusion::update(const vec3_t& z, const vec3_t& Bi, float sigma) {
    vec4_t q(x0);
```

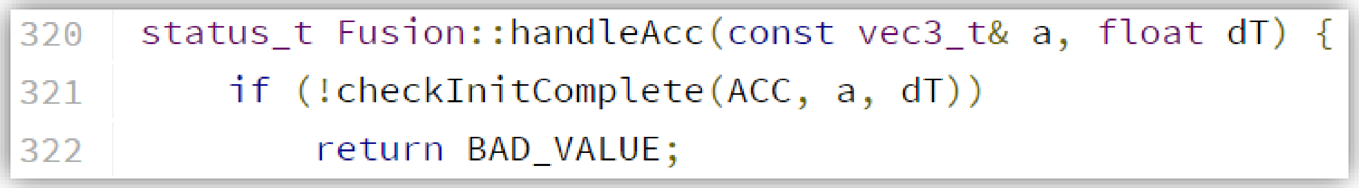
```
    const vec3_t e(z - Bb);
    const vec3_t dq(K[0]*e);

    q += getF(q)*(0.5f*dq);
    x0 = normalize_quat(q);
```

Source: https://android.googlesource.com/platform/frameworks/native/+/master/services/sensorservice/Fusion.cpp

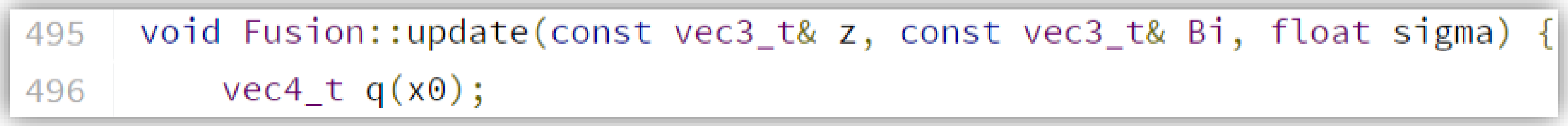
Claim 10

generating the **orientation output** based on the **first signal set**, the second signal set and the rotation output or based on the **first signal set** and the second signal set;

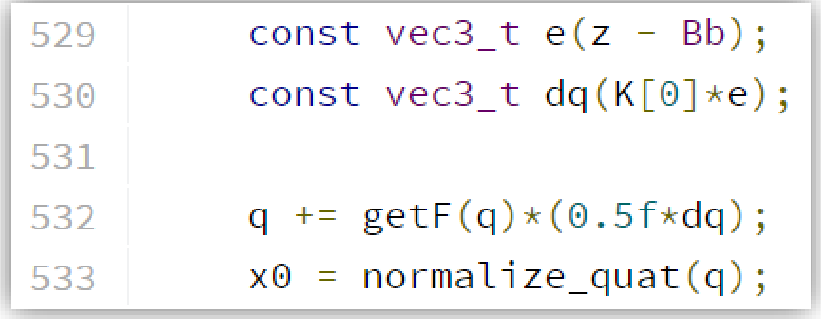
The `handleAcc()` function passes the accelerometer measurements (**first signal set**) `a` to the `update()` function, which updates the **orientation output** `x0`.



```
status_t Fusion::handleAcc(const vec3_t& a, float dT) {
    if (!checkInitComplete(ACC, a, dT))
        return BAD_VALUE;
```



```
void Fusion::update(const vec3_t& z, const vec3_t& Bi, float sigma) {
    vec4_t q(x0);
```



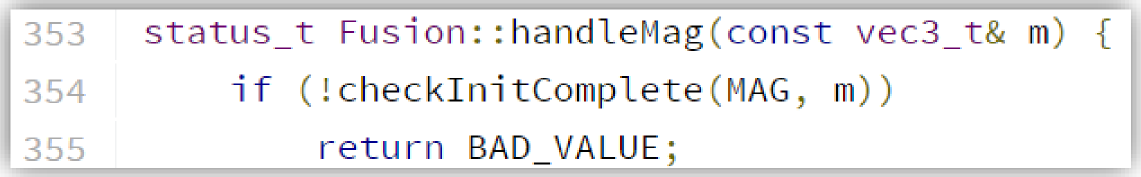
```
    const vec3_t e(z - Bb);
    const vec3_t dq(K[0]*e);

    q += getF(q)*(0.5f*dq);
    x0 = normalize_quat(q);
```

Source: https://android.googlesource.com/platform/frameworks/native/+/master/services/sensorservice/Fusion.cpp

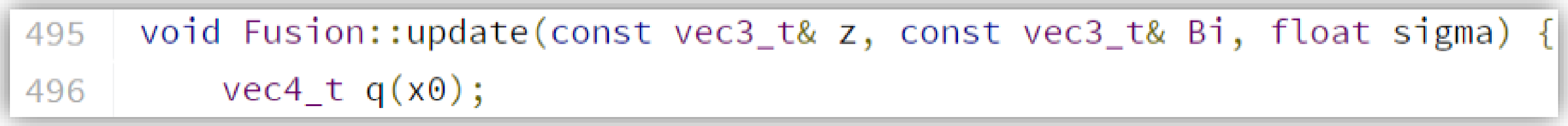
Claim 10

generating the **orientation output** based on the first signal set, the **second signal set** and the rotation output or based on the first signal set and the **second signal set**;

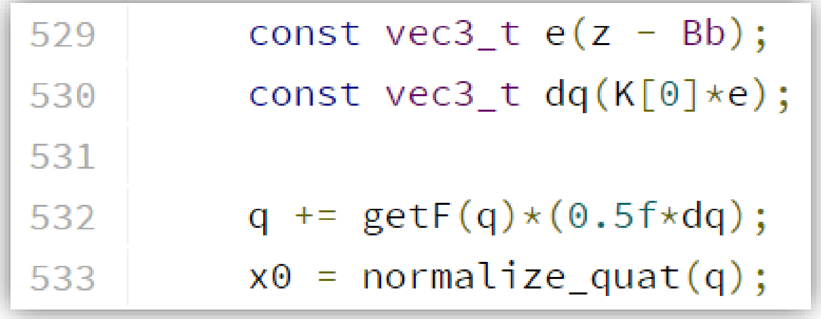
The `handleMag()` function passes the magnetometer measurements (**second signal set**) `m` to the same `update()`, which also updates the **orientation output** `x0`.



```
status_t Fusion::handleMag(const vec3_t& m) {
    if (!checkInitComplete(MAG, m))
        return BAD_VALUE;
```



```
void Fusion::update(const vec3_t& z, const vec3_t& Bi, float sigma) {
    vec4_t q(x0);
```

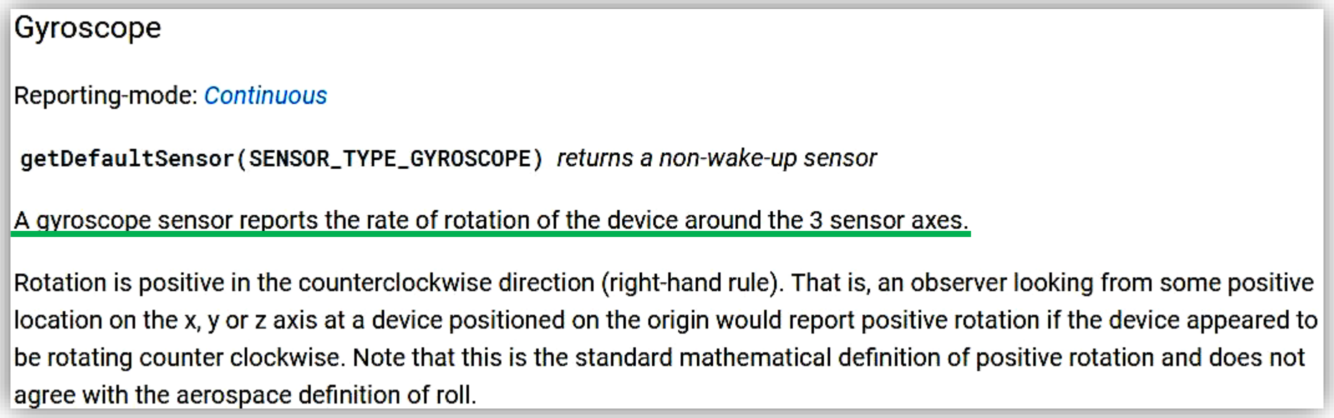


```
        const vec3_t e(z - Bb);
        const vec3_t dq(K[0]*e);

        q += getF(q)*(0.5f*dq);
        x0 = normalize_quat(q);
```

Source: https://android.googlesource.com/platform/frameworks/native/+/master/services/sensorservice/Fusion.cpp

Claim 10

generating a **rotation output** associated with a rotation of the 3D pointing device associated with three coordinate axes of a **spatial reference frame** associated with the 3D pointing device; and



Gyroscope

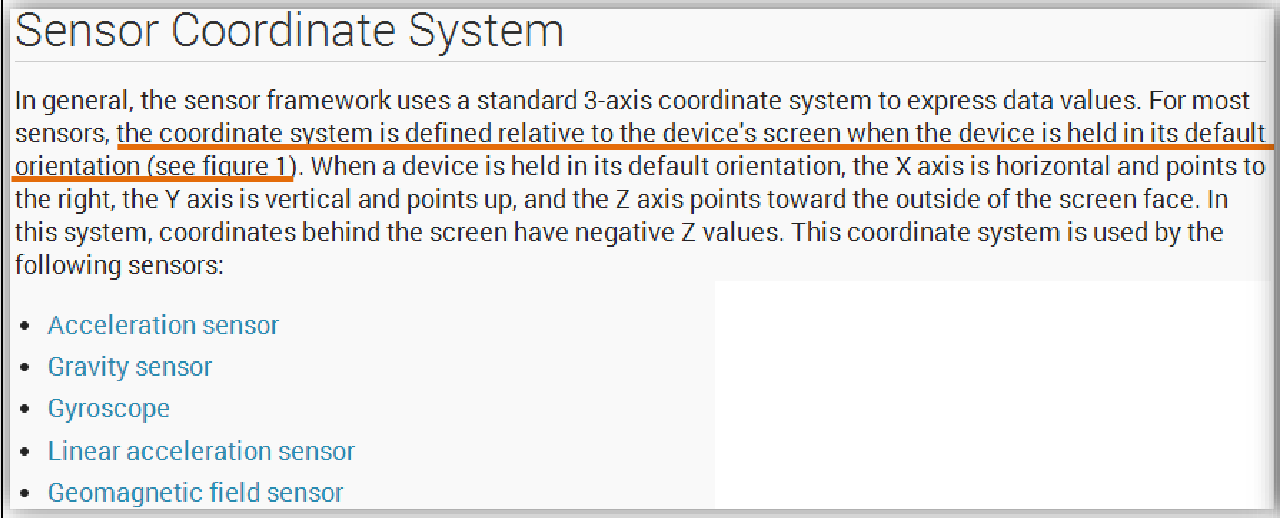
Reporting-mode: *Continuous*

`getDefaultSensor(SENSOR_TYPE_GYROSCOPE)` *returns a non-wake-up sensor*

A gyroscope sensor reports the rate of rotation of the device around the 3 sensor axes.

Rotation is positive in the counterclockwise direction (right-hand rule). That is, an observer looking from some positive location on the x, y or z axis at a device positioned on the origin would report positive rotation if the device appeared to be rotating counter clockwise. Note that this is the standard mathematical definition of positive rotation and does not agree with the aerospace definition of roll.

Source: https://source.android.com/devices/sensors/sensor-types#gyroscope



Sensor Coordinate System

In general, the sensor framework uses a standard 3-axis coordinate system to express data values. For most sensors, the coordinate system is defined relative to the device's screen when the device is held in its default orientation (see figure 1). When a device is held in its default orientation, the X axis is horizontal and points to the right, the Y axis is vertical and points up, and the Z axis points toward the outside of the screen face. In this system, coordinates behind the screen have negative Z values. This coordinate system is used by the following sensors:

- Acceleration sensor
- Gravity sensor
- Gyroscope
- Linear acceleration sensor
- Geomagnetic field sensor

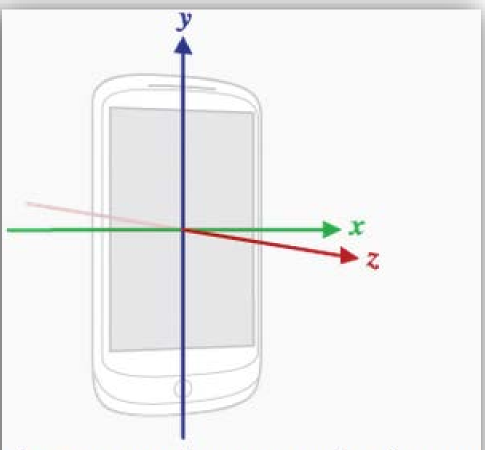


Figure 1. Coordinate system (relative to a device) that's used by the Sensor API.

Source: http://developer.android.com/guide/topics/sensors/sensors_overview.html#sensors-coords

SUBJECT TO CHANGE

Claim 10
using the orientation output and the rotation output to generate a transformed output associated with a **fixed reference frame** associated with a **display device** [<u>Court's construction: using the orientation output and the rotation output to generate a transformed output that corresponds to a two-dimensional movement in a plane that is parallel to the screen of a display device],</u>

The **fixed reference frame** is defined by the horizontal and vertical axes of pixels on HTC 10 **display device**.

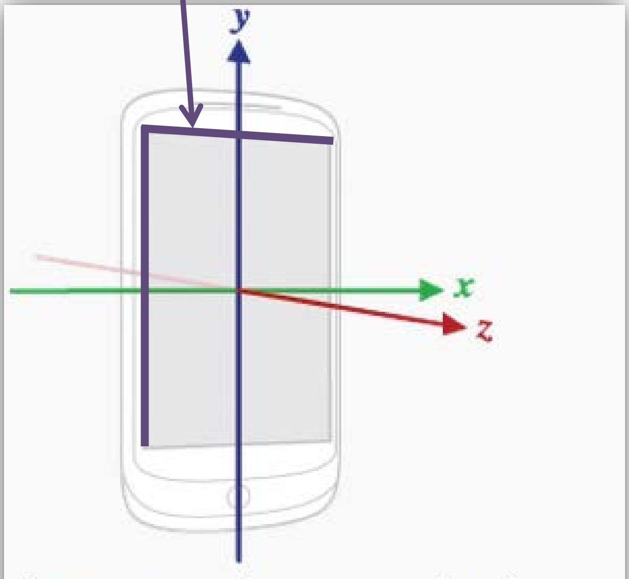
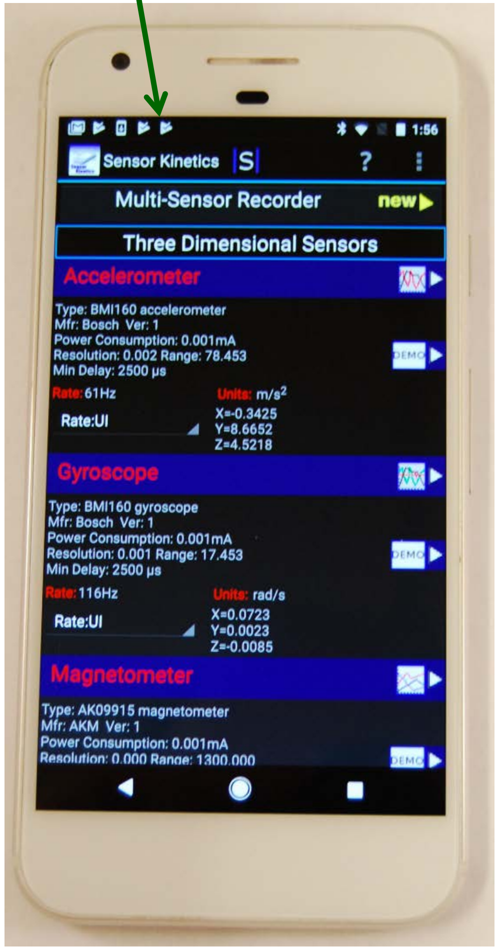


Figure 1. Coordinate system (relative to a device) that's used by the Sensor API.



Multi-Sensor Recorder

Three Dimensional Sensors

Accelerometer

Type: BMI160 accelerometer
Mfr: Bosch Ver: 1
Power Consumption: 0.001mA
Resolution: 0.002 Range: 78.453
Min Delay: 2500 µs
Rate: 61Hz
Units: m/s^2
Rate:UI
X=-0.3425
Y=8.6652
Z=4.5218

Gyroscope

Type: BMI160 gyroscope
Mfr: Bosch Ver: 1
Power Consumption: 0.001mA
Resolution: 0.001 Range: 17.453
Min Delay: 2500 µs
Rate: 116Hz
Units: rad/s
Rate:UI
X=0.0723
Y=0.0023
Z=-0.0085

Magnetometer

Type: AK09915 magnetometer
Mfr: AKM Ver: 1
Power Consumption: 0.001mA
Resolution: 0.000 Range: 1300.000

Source: http://developer.android.com/guide/topics/sensors/sensors_overview.html#sensors-coords

SUBJECT TO CHANGE

Claim 10

using the orientation output and the rotation output to generate a **transformed output** associated with a fixed reference frame associated with a **display device** [Court's construction: using the orientation output and the rotation output to generate a transformed output that corresponds to a two-dimensional movement in a plane that is parallel to the screen of a display device],

Google sells the Google Daydream View, which promotes and induces use of CyWee's patents. For example, the screen below shows a **transformed output** that corresponds to movements of the **Google Pixel** via AR and/or VR apps.

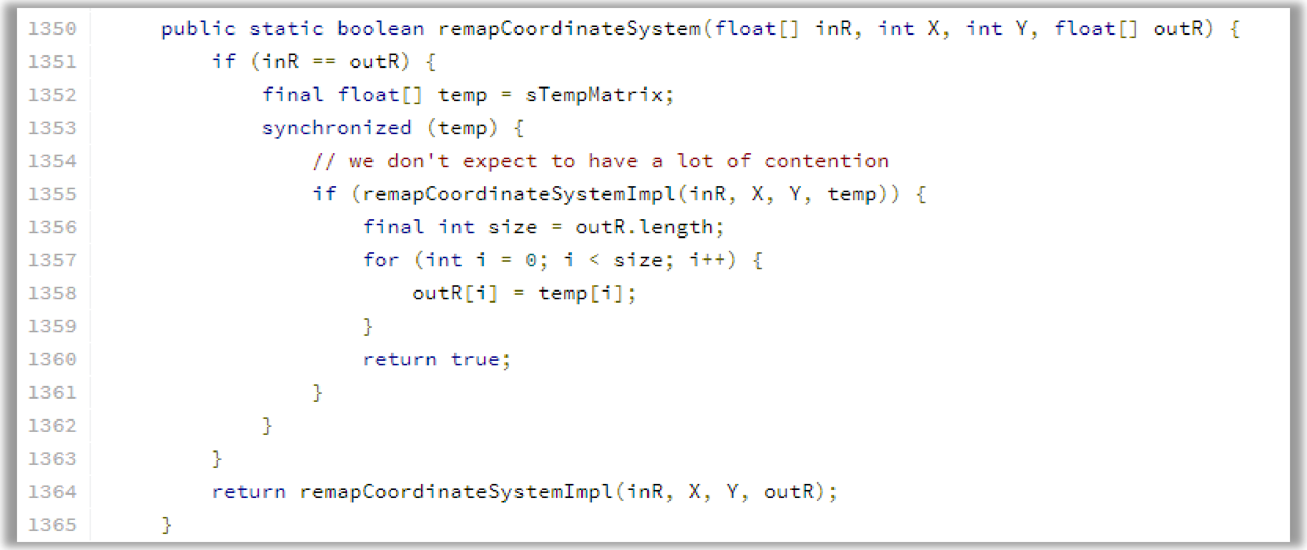
Source: https://vr.google.com/daydream/smartphonevr/

SUBJECT TO CHANGE

Claim 10

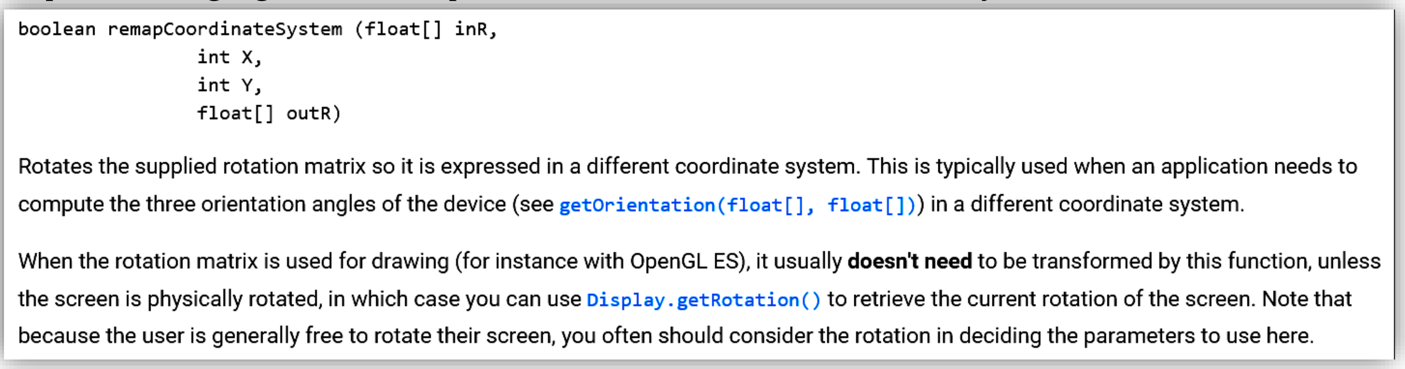
using the orientation output and the rotation output to generate a transformed output associated with a **fixed reference frame** associated with a **display device** [Court's construction: using the orientation output and the rotation output to generate a transformed output that corresponds to a two-dimensional movement in a plane that is parallel to the screen of a display device],

The `remapCoordinateSystem()` function transforms the orientation output (`inR`) to a transformed output (`outR`), associated with a two dimensional movement in a plane that is parallel to the screen of a **display device**.



```
    public static boolean remapCoordinateSystem(float[] inR, int X, int Y, float[] outR) {
        if (inR == outR) {
            final float[] temp = sTempMatrix;
            synchronized (temp) {
                // we don't expect to have a lot of contention
                if (remapCoordinateSystemImpl(inR, X, Y, temp)) {
                    final int size = outR.length;
                    for (int i = 0; i < size; i++) {
                        outR[i] = temp[i];
                    }
                    return true;
                }
            }
        }
        return remapCoordinateSystemImpl(inR, X, Y, outR);
    }
```

Source: https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/hardware/SensorManager.java



```
boolean remapCoordinateSystem (float[] inR,
                int X,
                int Y,
                float[] outR)
```

Rotates the supplied rotation matrix so it is expressed in a different coordinate system. This is typically used when an application needs to compute the three orientation angles of the device (see `getOrientation(float[], float[])`) in a different coordinate system.

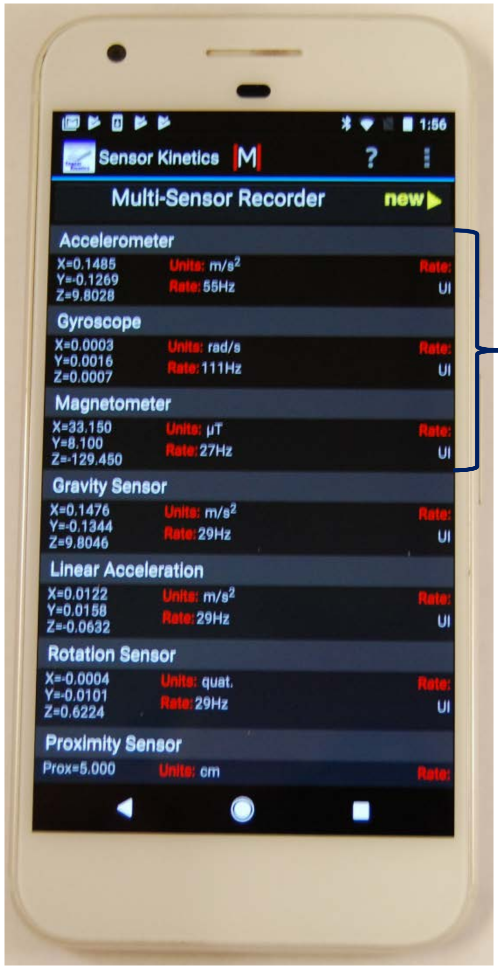
When the rotation matrix is used for drawing (for instance with OpenGL ES), it usually **doesn't need** to be transformed by this function, unless the screen is physically rotated, in which case you can use `Display.getRotation()` to retrieve the current rotation of the screen. Note that because the user is generally free to rotate their screen, you often should consider the rotation in deciding the parameters to use here.

Source: http://developer.android.com/reference/android/hardware/SensorManager.html

Claim 10

wherein the orientation output and the rotation output is generated by a **nine-axis motion sensor module**;

The Google Pixel includes a 3-axis gyroscope, a 3-axis accelerometer, and a 3-axis magnetometer which form a **nine-axis motion sensor module.**



Multi-Sensor Recorder

Accelerometer
X=0.1485 Y=-0.1269 Z=9.8028 Units: m/s² Rate: 55Hz Rate: UI

Gyroscope
X=0.0003 Y=0.0016 Z=0.0007 Units: rad/s Rate: 111Hz Rate: UI

Magnetometer
X=33.150 Y=8.100 Z=-129.450 Units: μT Rate: 27Hz Rate: UI

Gravity Sensor
X=0.1476 Y=-0.1344 Z=9.8046 Units: m/s² Rate: 29Hz Rate: UI

Linear Acceleration
X=0.0122 Y=0.0158 Z=-0.0632 Units: m/s² Rate: 29Hz Rate: UI

Rotation Sensor
X=-0.0004 Y=-0.0101 Z=0.6224 Units: quat. Rate: 29Hz Rate: UI

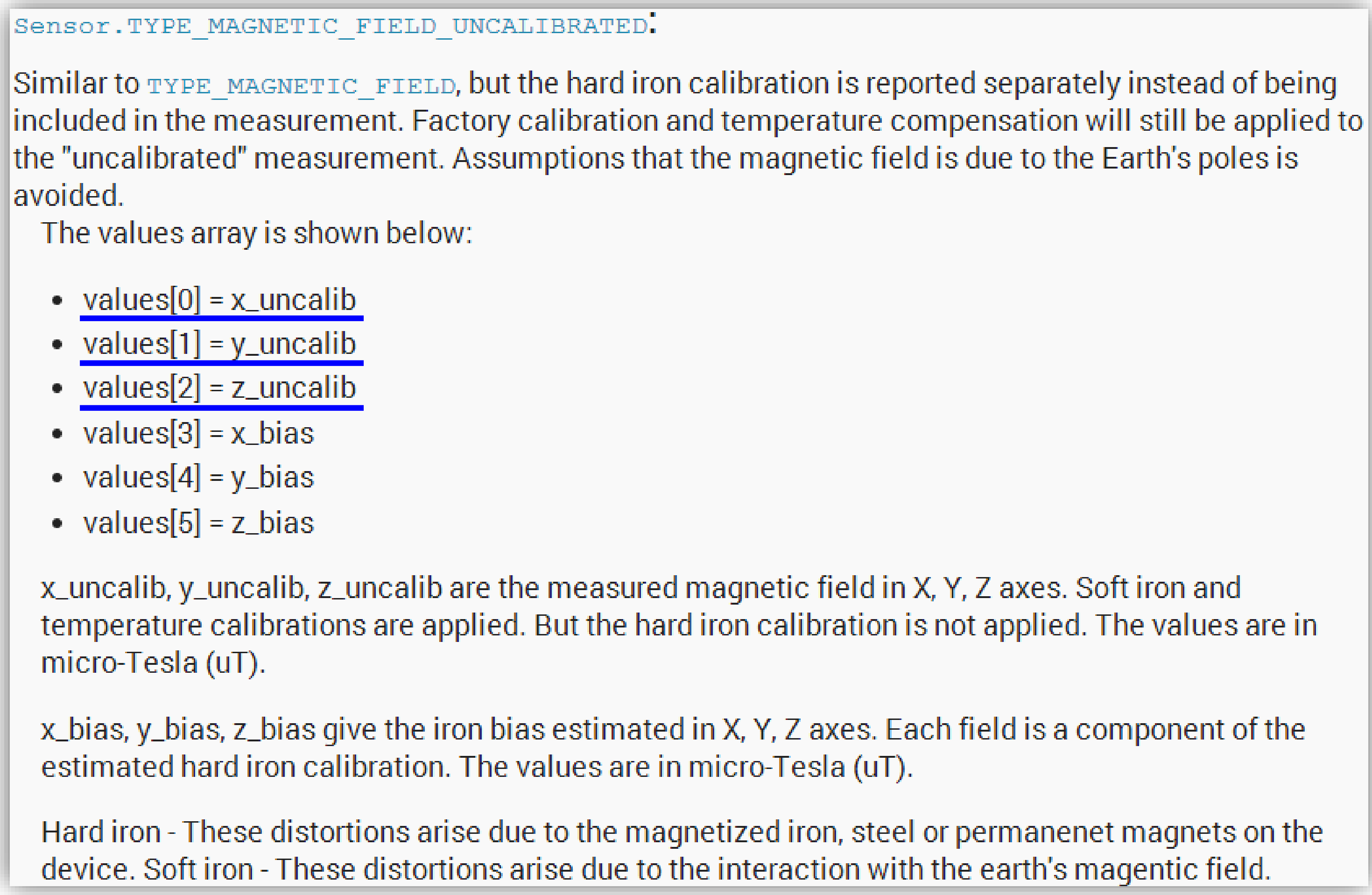
Proximity Sensor
Prox=5.000 Units: cm Rate:

SUBJECT TO CHANGE

Claim 10

obtaining one or more resultant deviation including a plurality of deviation angles using a plurality of **measured magnetisms Mx, My, Mz** and a plurality of predicted magnetism Mx′, My′ and Mz′ for the second signal set.

The **measured magnetisms Mx, My, Mz** are `values[0]-[2]`.



`Sensor.TYPE_MAGNETIC_FIELD_UNCALIBRATED`:

Similar to `TYPE_MAGNETIC_FIELD`, but the hard iron calibration is reported separately instead of being included in the measurement. Factory calibration and temperature compensation will still be applied to the "uncalibrated" measurement. Assumptions that the magnetic field is due to the Earth's poles is avoided.

The values array is shown below:

- values[0] = x_uncalib
- values[1] = y_uncalib
- values[2] = z_uncalib
- values[3] = x_bias
- values[4] = y_bias
- values[5] = z_bias

x_uncalib, y_uncalib, z_uncalib are the measured magnetic field in X, Y, Z axes. Soft iron and temperature calibrations are applied. But the hard iron calibration is not applied. The values are in micro-Tesla (uT).

x_bias, y_bias, z_bias give the iron bias estimated in X, Y, Z axes. Each field is a component of the estimated hard iron calibration. The values are in micro-Tesla (uT).

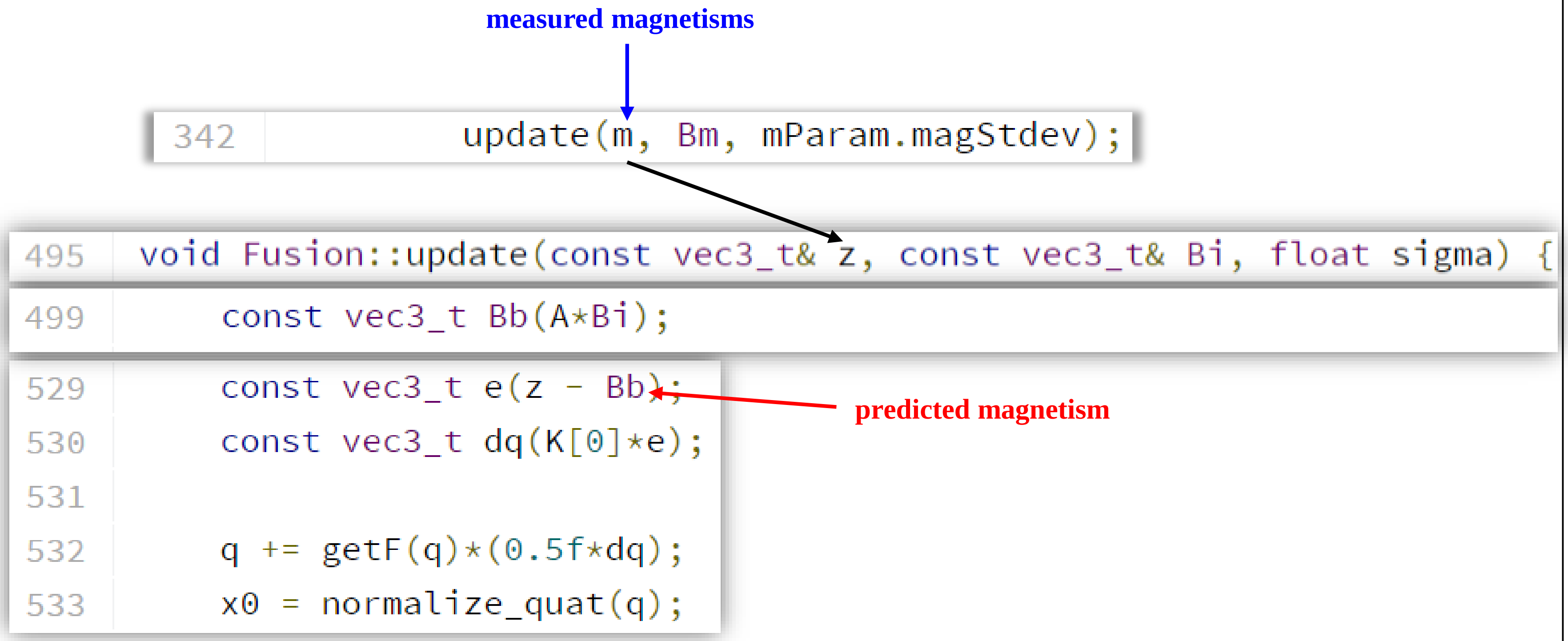
Hard iron - These distortions arise due to the magnetized iron, steel or permanenet magnets on the device. Soft iron - These distortions arise due to the interaction with the earth's magentic field.

Source: http://developer.android.com/reference/android/hardware/SensorEvent.html#values

Claim 10

obtaining one or more resultant deviation including a plurality of deviation angles using a plurality of **measured magnetisms Mx, My, Mz** and a plurality of **predicted magnetism** Mx′, My′ and Mz′ for the second signal set.

The **measured magnetisms**, z, and a **predicted magnetism**, Bb, are used to calculate a global variable x0 in quaternion form.



```
            update(m, Bm, mParam.magStdev);
```

```
void Fusion::update(const vec3_t& z, const vec3_t& Bi, float sigma) {
     const vec3_t Bb(A*Bi);
     const vec3_t e(z - Bb);
     const vec3_t dq(K[0]*e);

     q += getF(q)*(0.5f*dq);
     x0 = normalize_quat(q);
```

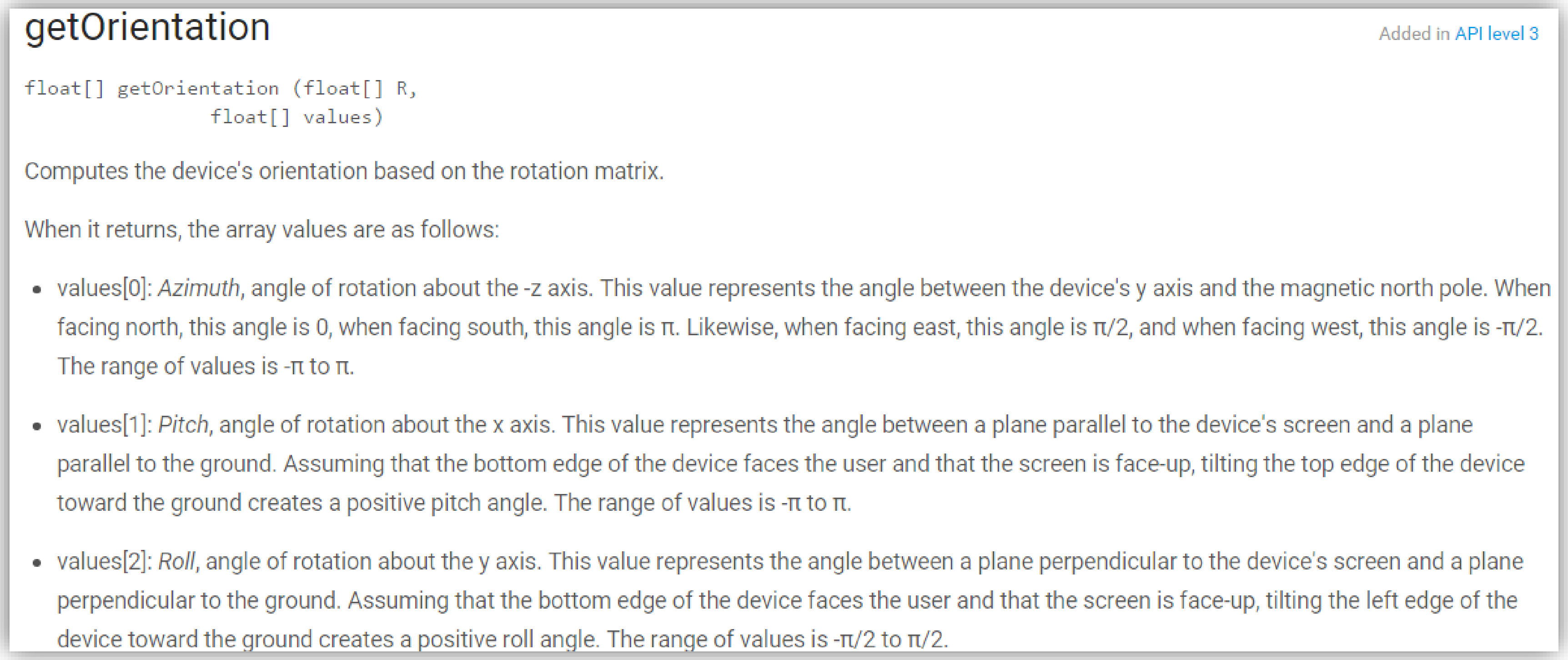
Source: https://android.googlesource.com/platform/frameworks/native/+/master/services/sensorservice/Fusion.cpp

Claim 10

obtaining one or more **resultant deviation including a plurality of deviation angles** using a plurality of measured magnetisms Mx, My, Mz and a plurality of predicted magnetism Mx′, My′ and Mz′ for the second signal set. .

The global variable `x0` is in quaternion form, and can easily be converted to resultant angles.

According to Android's developer library, the `getOrientation()` function "computes the device's orientation based on the rotation matrix," and returns **deviation angles** including the Azimuth, Pitch, and Roll angles.



getOrientation

Added in API level 3

```
float[] getOrientation (float[] R,
                float[] values)
```

Computes the device's orientation based on the rotation matrix.

When it returns, the array values are as follows:

- values[0]: *Azimuth*, angle of rotation about the -z axis. This value represents the angle between the device's y axis and the magnetic north pole. When facing north, this angle is 0, when facing south, this angle is π. Likewise, when facing east, this angle is π/2, and when facing west, this angle is -π/2. The range of values is -π to π.
- values[1]: *Pitch*, angle of rotation about the x axis. This value represents the angle between a plane parallel to the device's screen and a plane parallel to the ground. Assuming that the bottom edge of the device faces the user and that the screen is face-up, tilting the top edge of the device toward the ground creates a positive pitch angle. The range of values is -π to π.
- values[2]: *Roll*, angle of rotation about the y axis. This value represents the angle between a plane perpendicular to the device's screen and a plane perpendicular to the ground. Assuming that the bottom edge of the device faces the user and that the screen is face-up, tilting the left edge of the device toward the ground creates a positive roll angle. The range of values is -π/2 to π/2.

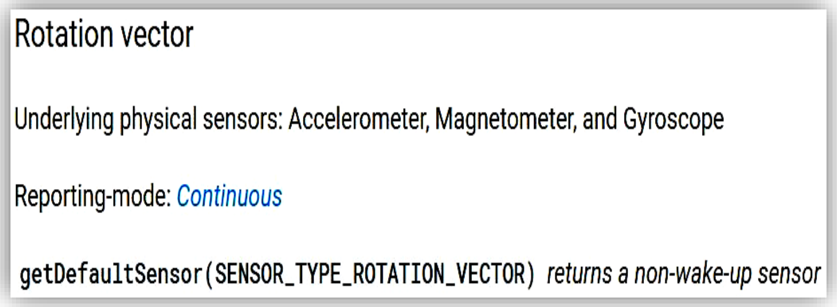
The `getRotationMatrixFromVector()` function "convert[s] a rotation vector to a rotation matrix," and the `getQuaternionFromVector()` function "convert[s] a rotation vector to a normalized quaternion." Therefore, the quaternion, `x0`, can be easily converted to its mathematically equivalent form, rotation matrix, and used by `getOrientation()` function to compute the orientation in its angular form.

Source: https://developer.android.com/reference/android/hardware/SensorManager.html#getOrientation(float[],%20float[])

SUBJECT TO CHANGE

Claim 10

obtaining one or more **resultant deviation including a plurality of deviation angles** using a plurality of measured magnetisms Mx, My, Mz and a plurality of predicted magnetism Mx′, My′ and Mz′ for the second signal set. .



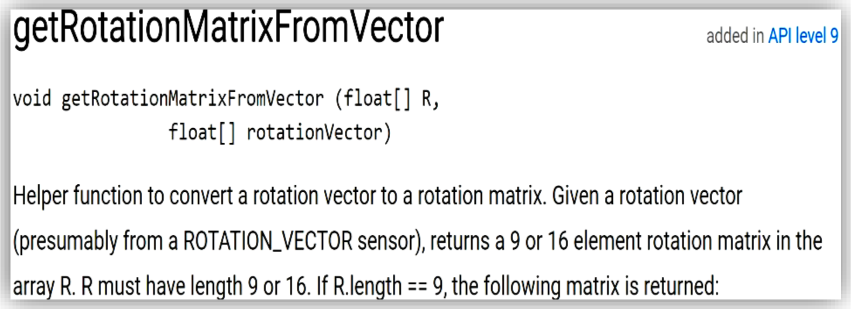
Rotation vector

Underlying physical sensors: Accelerometer, Magnetometer, and Gyroscope

Reporting-mode: Continuous

`getDefaultSensor(SENSOR_TYPE_ROTATION_VECTOR)` *returns a non-wake-up sensor*

Source: https://source.android.com/devices/sensors/sensor-types#rotation_vector

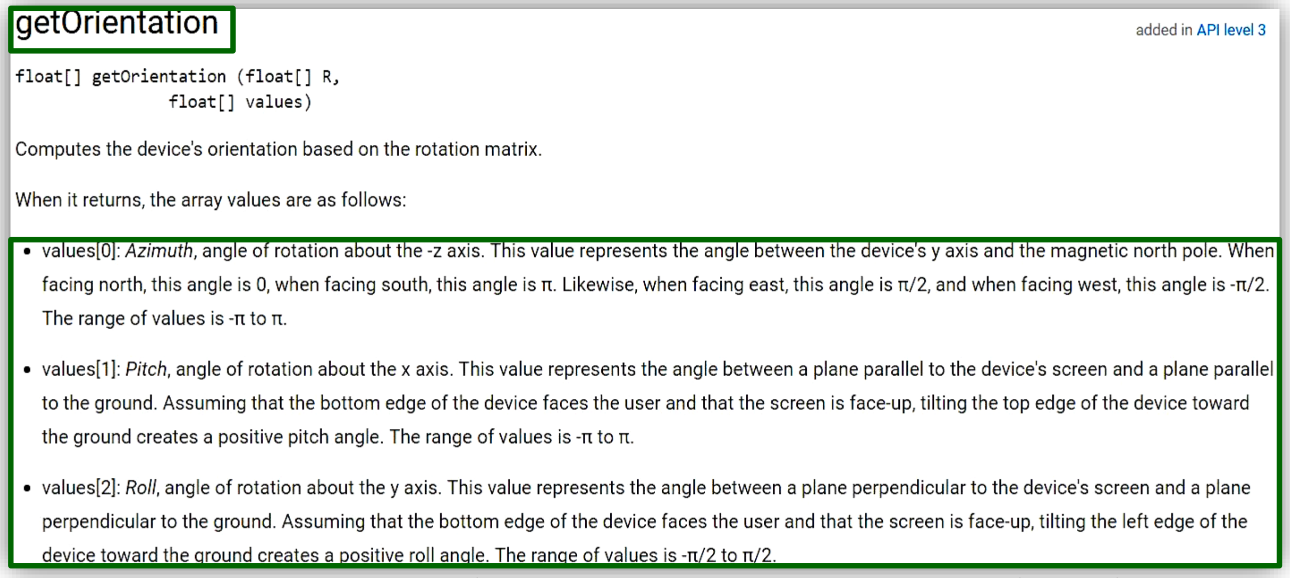


getRotationMatrixFromVector added in API level 9

```
void getRotationMatrixFromVector (float[] R,
                float[] rotationVector)
```

Helper function to convert a rotation vector to a rotation matrix. Given a rotation vector (presumably from a ROTATION_VECTOR sensor), returns a 9 or 16 element rotation matrix in the array R. R must have length 9 or 16. If R.length == 9, the following matrix is returned:

Source: https://developer.android.com/reference/android/hardware/SensorManager.html#getRotationMatrixFromVector(float[],%20float[])



getOrientation added in API level 3

```
float[] getOrientation (float[] R,
                float[] values)
```

Computes the device's orientation based on the rotation matrix.

When it returns, the array values are as follows:

- values[0]: *Azimuth*, angle of rotation about the -z axis. This value represents the angle between the device's y axis and the magnetic north pole. When facing north, this angle is 0, when facing south, this angle is π. Likewise, when facing east, this angle is π/2, and when facing west, this angle is -π/2. The range of values is -π to π.
- values[1]: *Pitch*, angle of rotation about the x axis. This value represents the angle between a plane parallel to the device's screen and a plane parallel to the ground. Assuming that the bottom edge of the device faces the user and that the screen is face-up, tilting the top edge of the device toward the ground creates a positive pitch angle. The range of values is -π to π.
- values[2]: *Roll*, angle of rotation about the y axis. This value represents the angle between a plane perpendicular to the device's screen and a plane perpendicular to the ground. Assuming that the bottom edge of the device faces the user and that the screen is face-up, tilting the left edge of the device toward the ground creates a positive roll angle. The range of values is -π/2 to π/2.

Source: https://developer.android.com/reference/android/hardware/SensorManager.html#getOrientation(float[],%20float[])

SUBJECT TO CHANGE